



UDM (Unified Data Manager)

An Open Source Time-series Data management Tool

Table of Contents

1	Executive Summary.....	6
2	Introduction	7
3	Overview of the system	9
3.1	Technical description of the internals.....	9
3.2	System setup.....	12
3.3	Operation	12
3.4	Current features.....	13
4	View data: Web Interface	16
4.1	How to Login into Jetspeed.....	16
4.2	How to Access a Jetspeed Portal	18
4.2.1	Case 1: Public Access.....	18
4.2.2	Case 2: Restricted Access (requirement: approved login account)	19
4.3	Create a Chart	20
4.3.1	Add a chart window to the screen.....	20
4.3.2	Adding Points:	22
4.3.3	Common Errors	26
5	Download Data from UDM	27
6	The Scripts Package.....	29
6.1	The *StartStop scripts	29
6.2	Automated startup at system startup (linux)	29
6.3	create_udm_home.sh.....	29
6.4	create_user.sh.....	30
6.5	backup.sh	30
7	PostgreSQL Installation	31
8	UDM Initial Setup.....	32
8.1	Unified Data Manager Setup for a Database Host.....	32
8.2	Unified Data Manager Setup for a Non-Database Host.....	32
8.3	How to Setup UDM Programs to Run at Windows StartUp.....	33
9	SysAdmin Tool.....	45
10	Jetspeed Installation	46

10.1	Download the JDBC driver	46
10.2	Download and Install Jetspeed	49
10.3	How to Use the Jetpeed Daemon	52
10.4	How to Create a Windows Service for Jetspeed	53
11	Jetspeed Setup	62
11.1	How to Create a Portal Space	62
11.2	How to Delete a Portal Space	64
11.3	How to Generate Various Security Settings in Jetspeed.....	65
11.4	How to create a page that only one user can edit, but everyone else can view	66
11.5	How to create a page for a user that only he can edit, but everyone else can view.....	67
11.6	How to create a page for a user that only he can view and edit	68
11.7	How to Insert a Company Logo into Jetspeed	76
11.7.1	How to convert an image or picture to a .GIF image or picture	77
11.7.2	How to insert a company logo into Jetspeed.....	77
11.7.3	How to change the header background color for the new header logo	79
12	Deploying UDM Portlets	81
12.1	High level instruction	81
12.2	Database Connection Pool (DBCP).....	81
13	Text File Data	84
13.1	The command	84
13.2	Parsing algorithm	84
14	XML Data.....	88
14.1	The command	88
14.2	Parsing algorithm	88
15	Modbus Data.....	94
15.1	Description.....	94
15.2	Point details	94
15.3	Installation	94
15.4	Operation.....	94
15.5	Troubleshooting.....	95
16	IPMI Data	97

16.1	Point details	97
17	RecentSweeper	98
18	UDM and BCVTB	99
18.1	Libraries to be downloaded	99
18.2	Setup work	99
18.3	Using the combined power of UDM and BCVTB.....	101
18.3.1	BACnet interface	101
18.3.2	BCVTB-UDM integration	103
18.3.3	Real time EnergyPlus in the BCVTB.....	104
18.3.4	BCVTB setup.....	107
19	Using UDM API	108
20	Analytics Container	126
20.1	Build the application	126
20.2	Create the Points	133
20.3	Edit the Points	134
20.4	Application deployment.....	134
21	Acknowledgements.....	135
22	References and Software Libraries	136
23	Appendix I: Ideal TDMS Features.....	137
23.1	Platform and browser support.....	137
23.2	Database	137
23.3	DataType support	137
23.4	Data collection	137
23.5	Time stamp	138
23.6	Metadata.....	138
	Source of metadata.....	138
23.7	Navigation	138
23.8	Data model.....	138
23.9	Security	138
23.10	Data query.....	138
23.11	Horizontal Scalability.....	138

23.12	User Interface.....	139
23.13	Extensibility	139
23.13.1	Analytics	139
23.13.2	Visualization	139
23.14	Data compression	139
23.15	Ease of deployment	139
23.16	Redundancy.....	140
23.17	Forecast data.....	140
23.18	Change management.....	140

1 Executive Summary

The buildings industry is a large industry with significant natural resource consumption. In order to make buildings more efficient, researchers in academia and national labs, building owners/operators, energy consultants and technology developers need to collect, organize and store measured and simulated time-series data. Currently, no tool is available that is appropriate for the work and is easily available to the users. LBNL proposes to fill that gap with Unified Data Manager (UDM).

The first version of UDM has already been released (<http://windows.lbl.gov/udm>) under a business-friendly modified OpenBSD license and is already in production at different sites. **The current capabilities of UDM include**

- 1) gathering data from majority of devices in the buildings industry – by reading csv file, reading xml file (including web service for both csv and xml), modbus and BACNet (Currently the BACNet capability is via BCVTB);
- 2) data storage in an industry standards based database (SQL),
- 3) a user-friendly, industry standards based and extensible web-based user interface
- 4) an easy way for 3rd party applications to exchange data with UDM (the data exchange API) and
- 5) the analytics container which helps users in executing and organizing custom applications

The basic framework has been built with extensibility in mind so that anybody can add features to it, whether for data collection, for metadata or for visualization.

2 Introduction

The cradle to grave energy consumption of built environment is as much as 65% of the total energy consumption of the country. Commercial and residential buildings alone, during their operations phase only, account for 38.9% of total U.S. energy consumption [1]. As far as carbon footprint is concerned, as per Brown *et al.* [2], residential and commercial buildings account for 39% of the carbon emissions in the United States. Building occupants use 13% of the total water consumed in the United States per day [1]. As of 2009, about 8% of U.S. energy demand goes to treating, pumping, and heating water [1]. One thing that is common across all the different types of facilities is that their operations are not very natural resource efficient.

Fortunately, this has not gone unnoticed and a lot of research is happening that attempts to make facility operations more natural resource efficient. Some broad examples of such initiatives are:

- Self-healing building systems
- Automated Demand Response
- Automated Fault Detection and Diagnostics
- Holistically Optimized Facility Operation (Integrated Building Controls)

However, in order to test these technologies, demonstration facilities need to be monitored and simulated. And for wide scale deployment of the technologies, a whole ecosystem of monitoring and simulation is required (hardware, software and workforce that implement and support the hardware and the software). Both monitoring and simulation generate time-series data. Please see Appendix I for the features necessary for time series data management systems. Unfortunately, there is no tool available, either commercial or open source, that satisfies all the criteria. Figure 2-1 is a pictorial representation of the need described above.

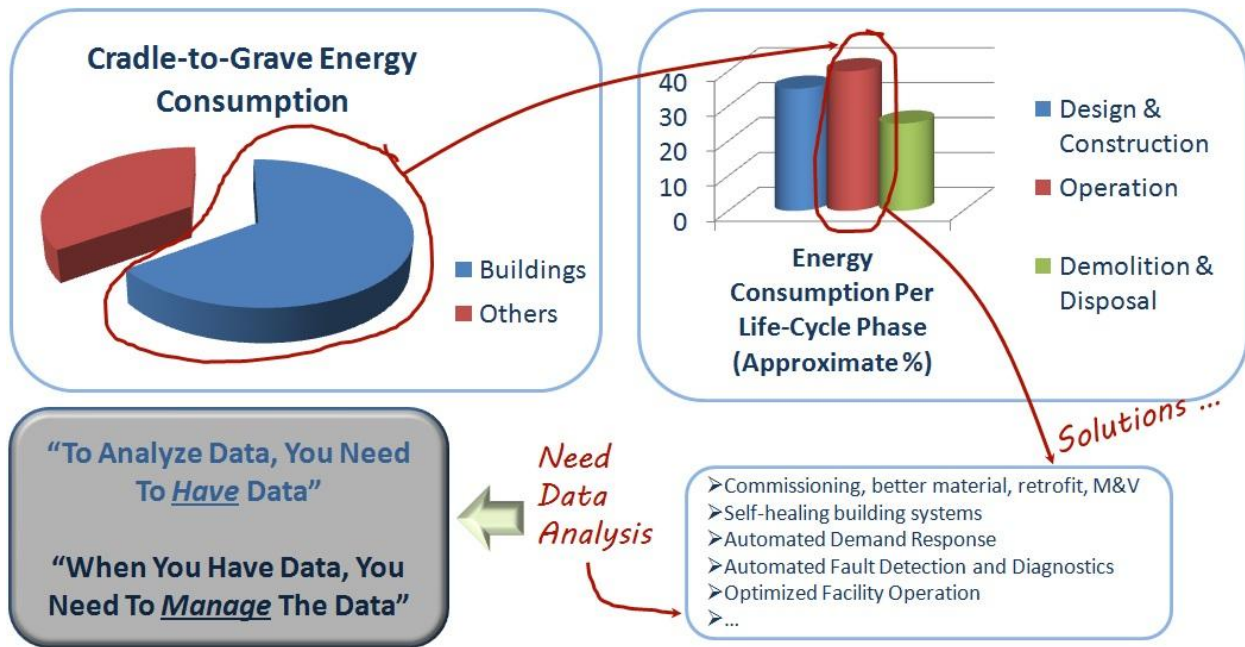


Figure 2-1: The need

3 Overview of the system

3.1 Technical description of the internals

Figure 3-1 shows the high level overview of UDM as it has been architected. Each of the six components addresses a specific problem area as described below.

- **Data Collection** – UDM supports a variety of industry standard communication protocols. The system is extensible such that any number of ProtocolCommunicators can be built and added to the system.
- **Data Storage** – secure and scalable storage mechanism that follows industry standards
- **Data Organization** – industry standards based meta-data structure (information about time-series data) - BIM based building component description, GIS based navigation etc.
- **Visualization**
 - User friendly and modular visualization to facilitate navigation through potentially large databases
 - Flexibility and extensibility so that users can build their own visualization screens in minutes
 - The extensibility of the system also allows addition of new chart types.
- **Data Exchange** – a standards based API, that allows 3rd party applications to programmatically pull data from the system, perform calculations outside of UDM and finally write data back to the UDM database. A few examples of such applications are simulation programs, Building Controls Virtual Test Bed (BCVTB), Fault Detection and Diagnostics (FDD) tools and optimization tools.
- **Data Analysis** – a framework within UDM, where users can build their own applications, run them (continuously or in batch mode) and organize them. Sample analyses are: demand response, FDD, optimization, model calibration etc.

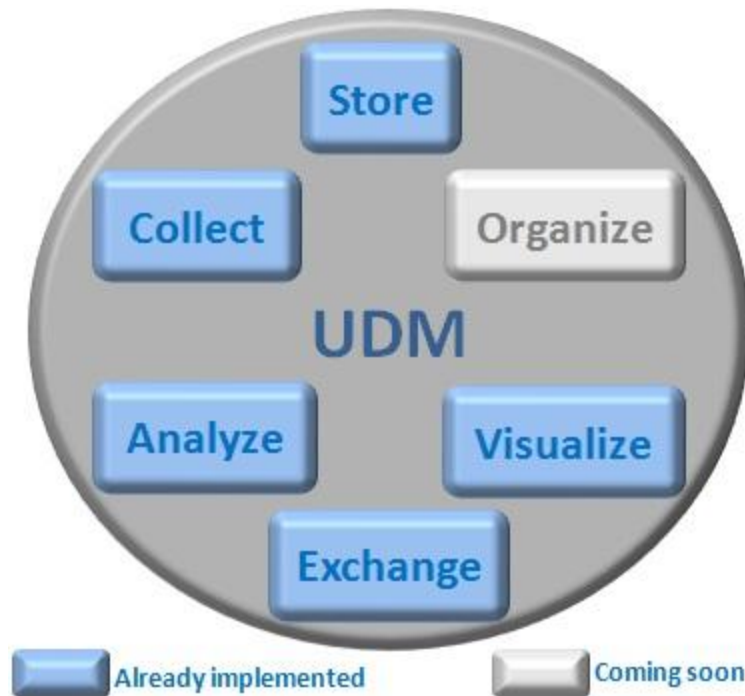


Figure 3-1: High level overview of the system

Figure 3-2 shows a simplified schematic diagram of the system as can be deployed in the field. A few examples of *Data Source Type* are xml files, structured csv files, external database, bacnet, modbus, SNMP, XML web services etc. Currently, *ProtocolCommunicator App* for modbus, csv, xml and IPMI are implemented. The backend database and the ProtocolCommunicator app can reside either on the same host or on different hosts. We decided to use a database that conforms to the SQL standard because the support ecosystem for these databases is widely available. As our first attempt, we used the postgresSQL database program as the backend database. We kept our SQL statements as standard-oriented as possible. It should be noted here that UDM's database structure is very different from the way usually Relational Databases are used for time-series data.

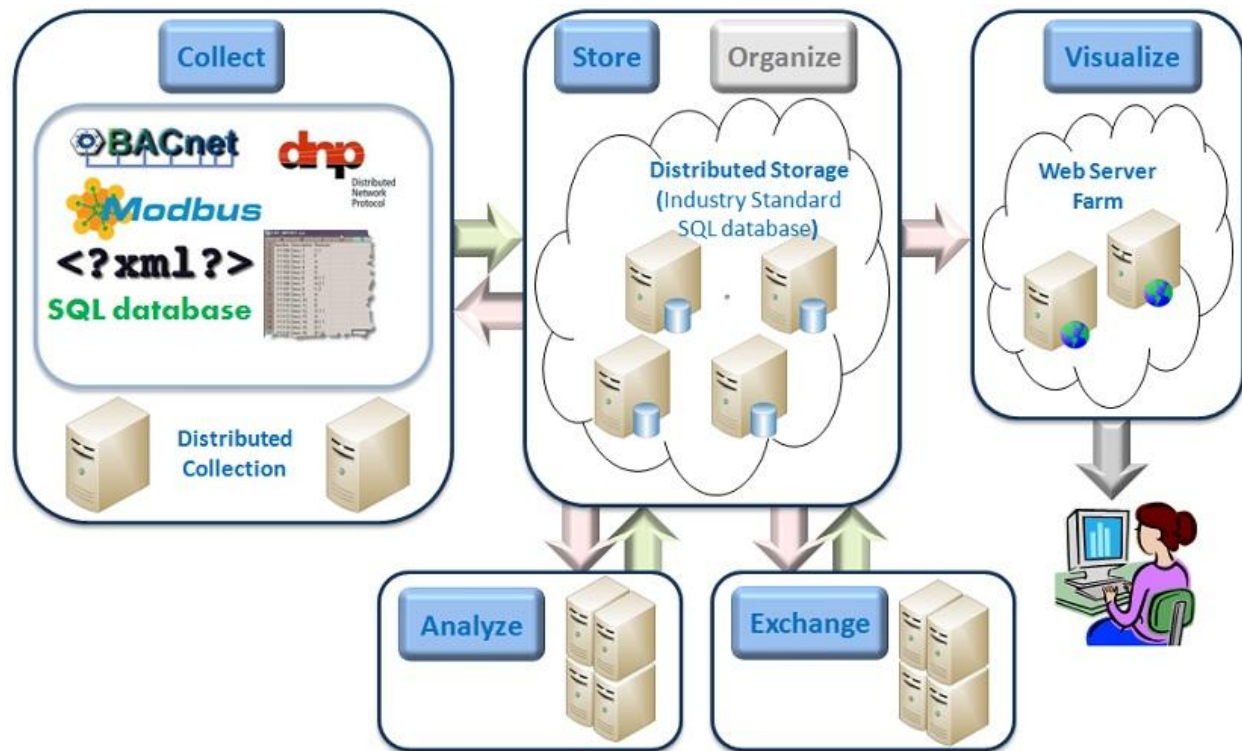


Figure 3-2: The system as can be deployed in the field

Every time-series data stream is a different Point in the system. In its current configuration, only a limited amount of metadata (information about the time-series data) can be stored in the database. Right now all that information is stored in the database in a **table called Point**. Each row in that table is dedicated to a single time-series point. The columns contain information about the unique identifier for a point, the human readable description of it, the unique identifier for the data source type, how to gather data for this point from the data source, how to interpret that data, whether the data is forecast data and how often the data source should be polled to collect the data for this point. The time-series data is stored in tables dedicated for individual points. The mapping between the points and their *DataTables* is maintained in a **table called Point_DataTable_Map**. Every time a point is created in the system a new datatable, dedicated to that point, is created in the database. Similarly, when a point is deleted from the system, all datatables associated with it are dropped from the database.

All the applications were built on the java platform to make the system compatible with both Windows and *nix operating systems.

Since the database structure is not very flat, making changes to the system using plain SQL statements can prove to be cumbersome. Therefore a system administration utility has been developed which takes a csv file as its input and outputs another csv file if there is supposed to be any output. The operations supported at present include point creation, point deletion, point editing and point import. The first three are probably self-explanatory; the point import action imports all the details about all the points into a csv file.

3.2 System setup

- 1) Since all the applications are java-based, the first step in setup is to install Java Runtime Environment (JRE) on the host computer.
- 2) The next thing is to install the database program. It is postgresSQL, for the current version of the system. On Windows OS, the installation is fairly simple. For linux, a script has been created. It is available in the package called scripts.
- 3) Then the setup tool contained within the udmDatabaseSetup.zip package needs to be executed. It includes a few text files with the database configuration information and a java package which performs the actual task of setting up the initial database structure.
- 4) The steps described above constitute the generic part of the setup process. After that, the site-specific setup needs to be performed. Essentially, it involves setting up every time-series point in the system with the individual details. A spreadsheet is created using specific headers. Each row of that spreadsheet contains information for a single point. The Point specific information depends on the particular ProtocolCommunicator to which the Point belongs. The spreadsheet is then converted to a csv file which is used as the input to the system administration tool. Each ProtocolCommunicator package, such as udmModbus.zip or udmTextFile.zip includes sample .csv files. The column headers of the spreadsheet are essentially mapped to the columns in the Point table in the database.

	B	C	D	E	F	G	H	I
1	description	datasourceprotocolid	datasourceaddress1	datas	datasourceaddress3	datas	datasourcedetail1	datasourcedetail2
2	DAVIS_temperature	2	http://www.wrh.noaa.gov/forecast/xml/xml.php?duration=168&interval=1&lat=38.5449065&lon=-121.7405167	download		parent<parent<pare	EEE MMM dd HH:mm:ss yyyy z;MM	
3	DAVIS_dewpoint	2	http://www.wrh.noaa.gov/forecast/xml/xml.php?duration=168&interval=1&lat=38.5449065&lon=-121.7405167	download		parent<parent<pare	EEE MMM dd HH:mm:ss yyyy z;MM	
4	DAVIS_rh	2	http://www.wrh.noaa.gov/forecast/xml/xml.php?duration=168&interval=1&lat=38.5449065&lon=-121.7405167	download		parent<parent<pare	EEE MMM dd HH:mm:ss yyyy z;MM	
5	DAVIS_skyCover	2	http://www.wrh.noaa.gov/forecast/xml/xml.php?duration=168&interval=1&lat=38.5449065&lon=-121.7405167	download		parent<parent<pare	EEE MMM dd HH:mm:ss yyyy z;MM	
6	DAVIS_windSpeed	2	http://www.wrh.noaa.gov/forecast/xml/xml.php?duration=168&interval=1&lat=38.5449065&lon=-121.7405167	download		parent<parent<pare	EEE MMM dd HH:mm:ss yyyy z;MM	
7	DAVIS_windDirection	2	http://www.wrh.noaa.gov/forecast/xml/xml.php?duration=168&interval=1&lat=38.5449065&lon=-121.7405167	download		parent<parent<pare	EEE MMM dd HH:mm:ss yyyy z;MM	
8	DAVIS_pop	2	http://www.wrh.noaa.gov/forecast/xml/xml.php?duration=168&interval=1&lat=38.5449065&lon=-121.7405167	download		parent<parent<pare	EEE MMM dd HH:mm:ss yyyy z;MM	

Figure 3-3: Setting up the point details in a spreadsheet for exporting to the database

- 5) After the point creation, the ProtocolCommunicator applications need to be started. The ProtocolCommunicator apps run as services; thus, if they are set up to start up automatically after a system start-up, then they will run forever without any further manual intervention.

3.3 Operation

Every instance of a ProtocolCommunicator app has a unique identifier associated with it. At the time of start-up, a ProtocolCommunicator app queries the database for points that match the instance's unique identifier. The details of the points are loaded into the memory. Then it runs in an infinite loop. It

“sleeps” until the next poll time arrives and then performs the work of data fetching, parsing and storing and then again goes back to sleep until the next poll time.

3.4 Current features

1. Optional **file download** capability. For example, if the data is published in a csv file or an XML on a website or on any remote host, the file can be fetched to the local host.
2. Optional **unzip** capability. For example, California ISO publishes its demand and price data in a zipped file format. One then has to unzip the file to get the actual csv or XML file. Unzipping might be required for locally residing files as well.
3. **Generic XML file parsing**. For example, we are collecting 7 day hourly weather forecast data from NOAA website in an XML file, parsing the file for values and timestamps for different quantities.
4. Structured **csv or tab separated file parsing**.
5. Support for data collection using **Modbus/TCP** protocol
6. Support for IPMI protocol
7. Capability to store **forecast data**. Forecast data is different from actual historic data in terms of timestamp. In historic data, there is only one timestamp associated with a value. In forecast data, there is a timestamp for when the forecast was made and a second timestamp for the time for which the forecast was made. For example, weather forecast for Sunday, is made on Thursday, Friday and Saturday. We should capture both notions of time for the sake of clarity, correctness and possible analysis of the performance of the forecast.
8. Timestamp is stored in **UTC millisecond**. This ensures transparent operation across time zones.
9. **Automatic application of offset on the poll time**. For example, if the poll interval for a point is 1 hour, the polling takes place slightly after the top of the hour. This helps in smoothing the load on the local host. But more importantly, it helps in avoiding time-outs and lack-of-response from a remote server where several people send request at the top of the hour (such as CA ISO oasis server).
10. **Backfill mode** for the text file ProtocolCommunicator app.
11. **Multi-threaded** operation of the data collector apps enables better performance and scalability. A thread can have one or more points for which it performs data collection. In the current version, a separate thread is created for every different combination of address of the data-source (IP address or web URL) and the poll interval.
12. Backfill does not require any manual intervention, such as making manual changes in datatable start time (archiving start time).
13. Basic data serving capability using the API
14. A portlet-based (that is, easily and highly customizable) visualization of time-series data. Figure 3-4 (searching for points in the database) and Figure 5 (a dashboard being built in 2 minutes) attempt to provide a glimpse of the visualization in its current state.

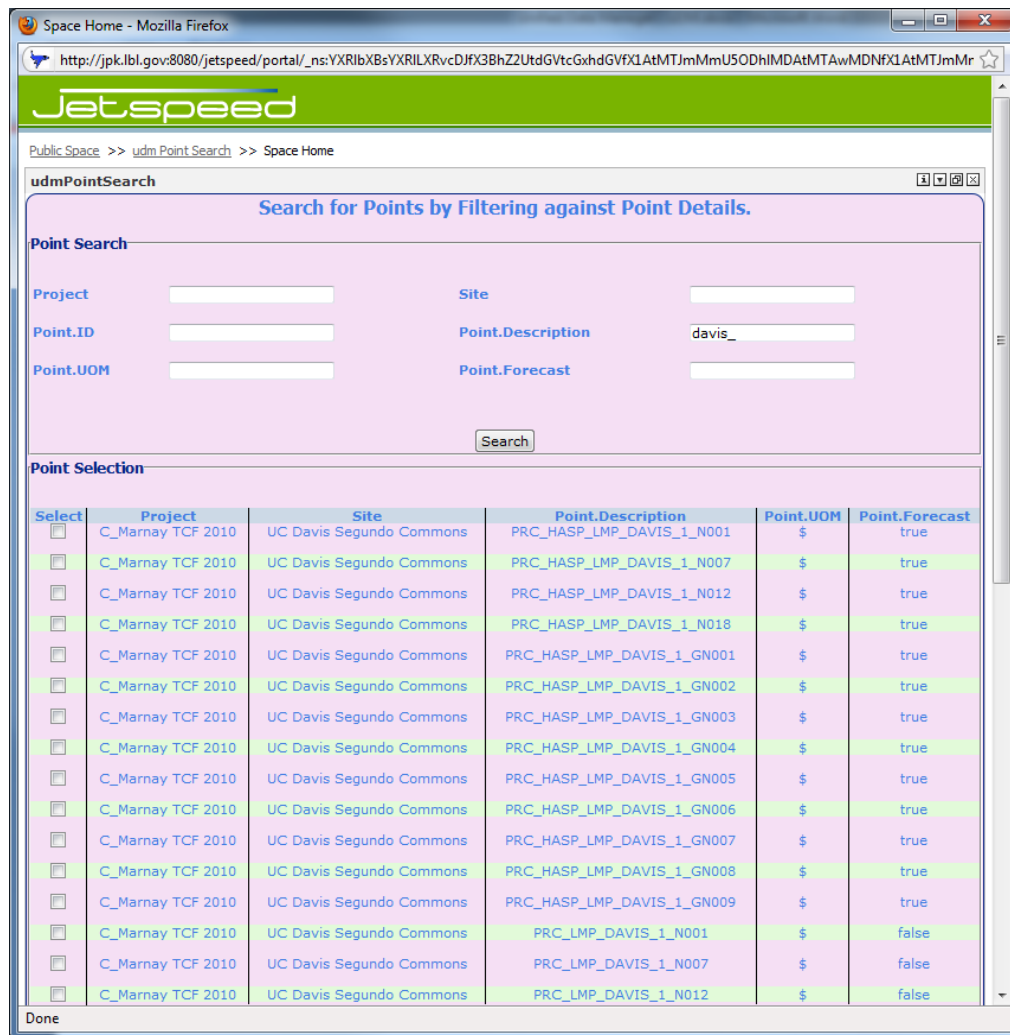


Figure 3-4: Screen where users search for points in the database



Figure 3-5: A dashboard being built

4 View data: Web Interface

4.1 How to Login into Jetspeed

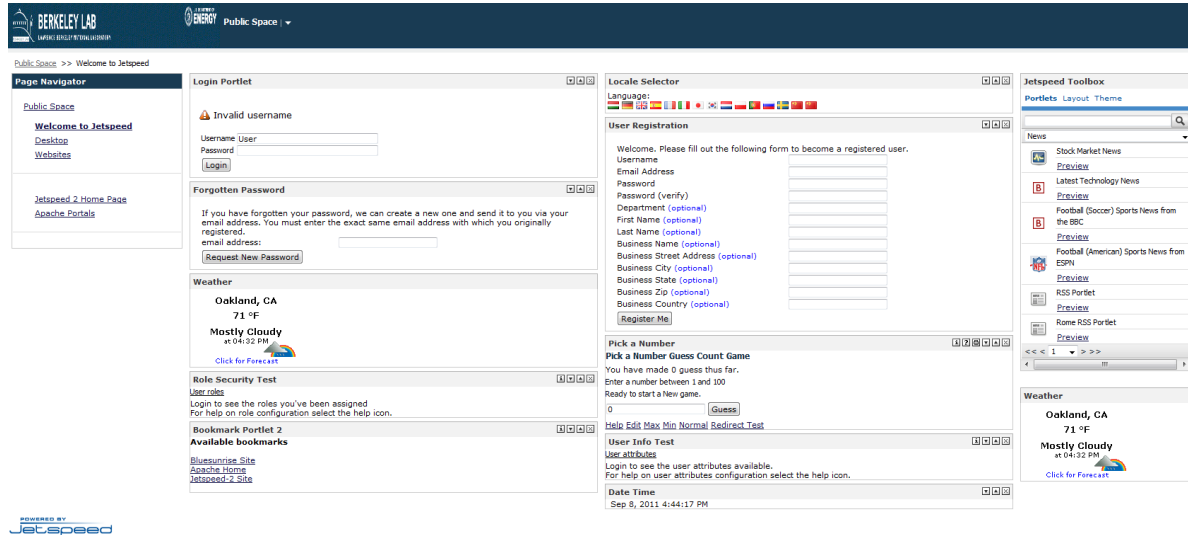
1. Enter your username and password as shown in the image below. Then, click on “Login.”

The screenshot displays the Jetspeed web interface. At the top, there is a header with the Berkeley Lab logo and 'Public Space' link. Below the header, the main content area is divided into several sections:

- Page Navigator:** Contains links for 'Public Space', 'Welcome to Jetspeed', 'Desktop', 'Websites', 'Jetspeed 2 Home Page', and 'Apache Portals'.
- Login Portlet:** Features a 'Username' field (containing 'User2'), a 'Password' field (masked with dots), a 'Login' button, a 'Forgotten Password' link, and a 'Request New Password' button. Below this is a 'Weather' section for 'Oakland, CA' showing '71 °F' and 'Mostly Cloudy'.
- Locale Selector:** A dropdown menu for selecting a language.
- User Registration:** A form for new users to register, including fields for 'Username', 'Email Address', 'Password', 'Password (verify)', 'Department (optional)', 'First Name (optional)', 'Last Name (optional)', 'Business Name (optional)', 'Business Street Address (optional)', 'Business City (optional)', 'Business State (optional)', 'Business Zip (optional)', and 'Business Country (optional)'. It also includes a 'Register Me' button.
- Role Security Test:** A section for users to see their assigned roles.
- Bookmark Portlet 2:** A section for available bookmarks, including 'Bluesunrise Site', 'Apache Home', and 'Jetspeed-2 Site'.
- Pick a Number:** A section for a 'Number Guess Count Game' where users can enter a number between 1 and 100.
- User Info Test:** A section for users to see their attributes.
- Date Time:** A section showing the current date and time.
- Jetspeed Toolbox:** A sidebar on the right containing various tools and links, including 'News', 'Stock Market News', 'Latest Technology News', 'Football (Soccer) Sports News from the BBC', 'Football (American) Sports News from ESPN', 'RSS Portlet', and 'Rome RSS Portlet'.

2. If you have successfully logged in, you will see a page similar to the page in the image shown below.

Error 2: invalid username

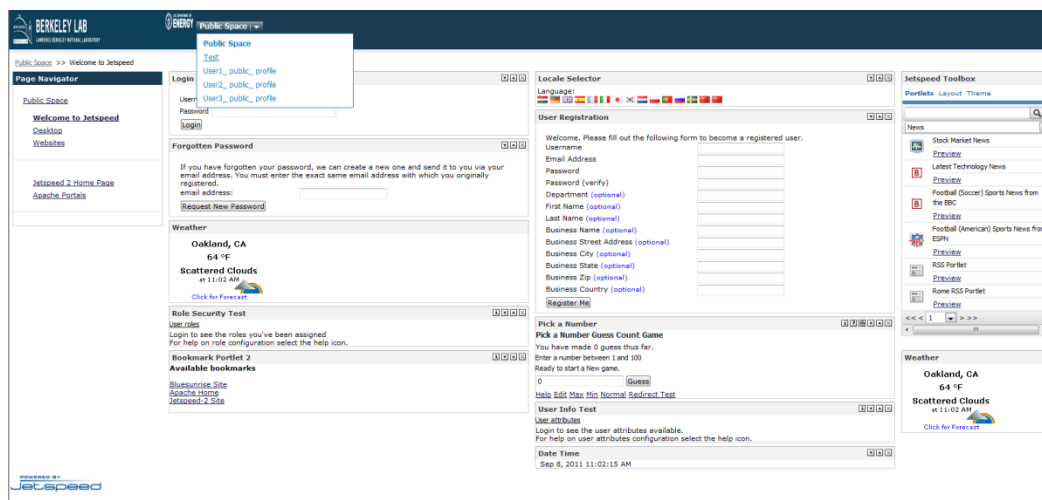


4.2 How to Access a Jetspeed Portal

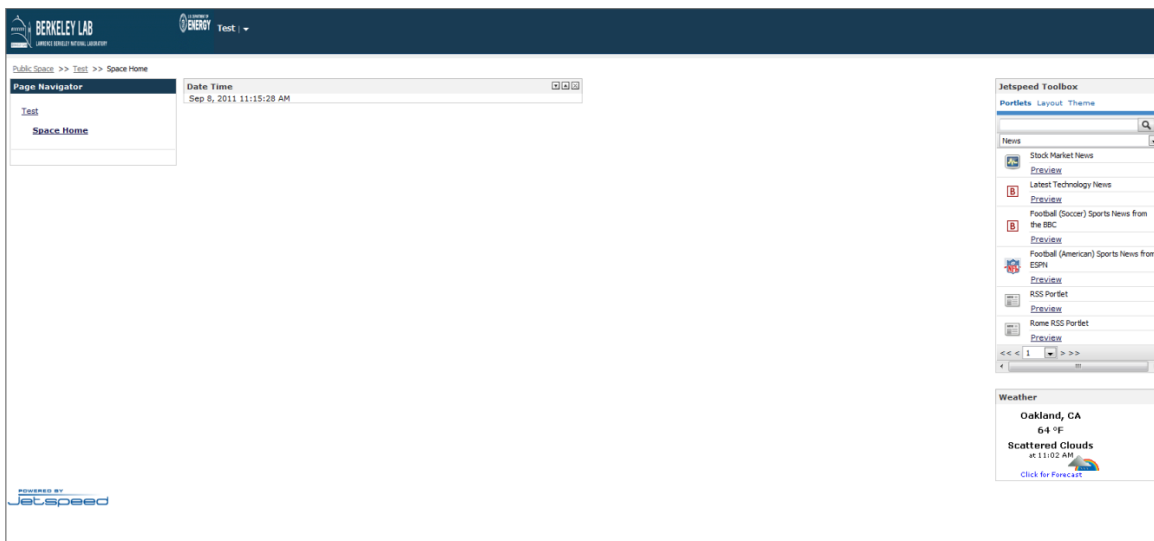
There are two ways in which a Jetspeed portal can be accessed by a user:

4.2.1 Case 1: Public Access

Go to the main menu (the drop down menu shown in the image below) and select the portal from the list. As seen in the example below, the user is selecting the portal, Test, by clicking on Test from the drop down menu



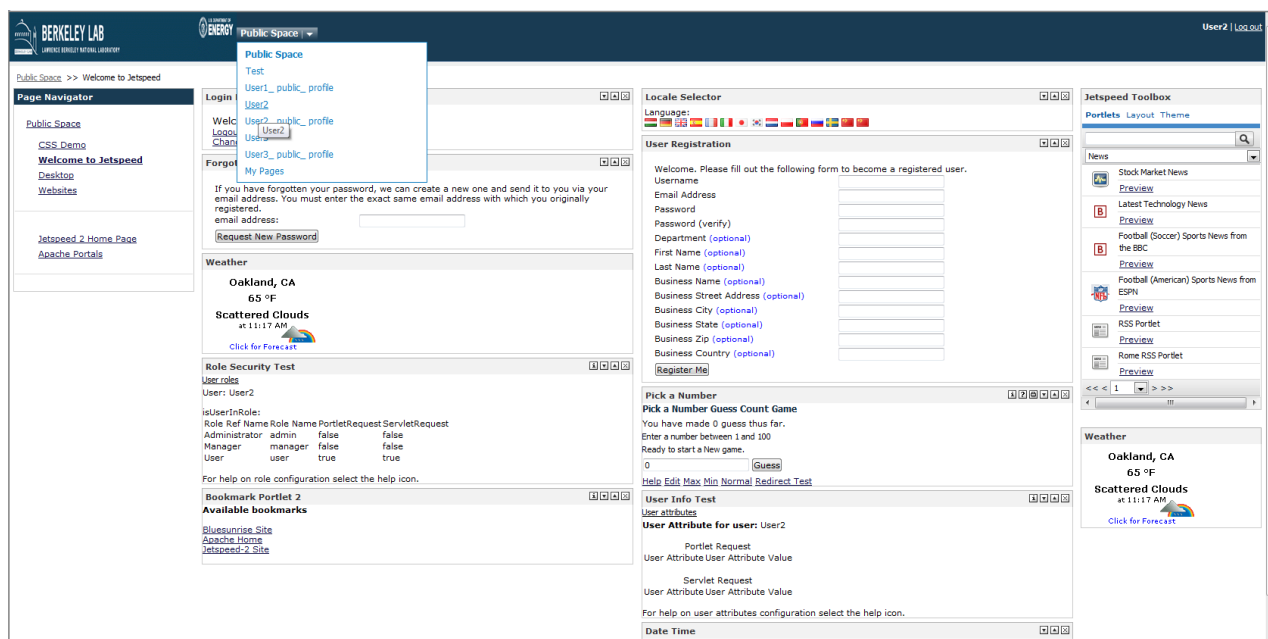
Once the user has selected the portal, he will be taken to the portal's main page. As seen in the image below, the user has gone to Test's main page.



4.2.2 Case 2: Restricted Access (requirement: approved login account)

Without logging in, the portal will not be visible to the user. Once the user has logged in, he will be able to see and access the portal. In the example shown in the image below, the user, user2, has logged in and now has access to the following portals:

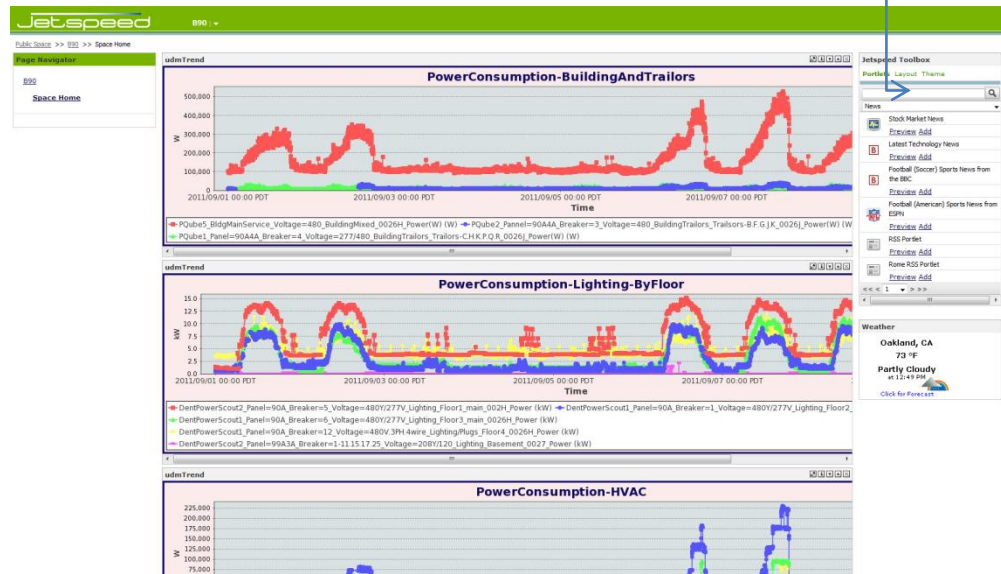
- the portals that the user, from case 1, had access to
- the portals that only approved logins have access to.



4.3 Create a Chart

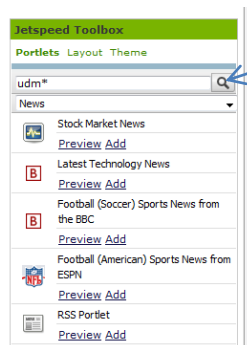
4.3.1 Add a chart window to the screen

Jetspeed search engine

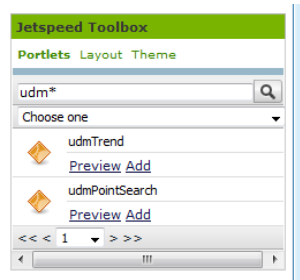


1. Enter **udm*** into the Jetspeed search engine.

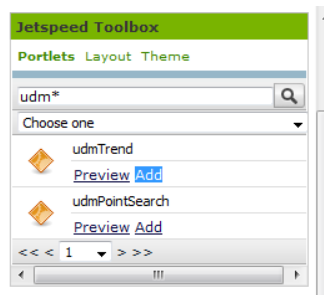
2. Then, click on the **search** button.



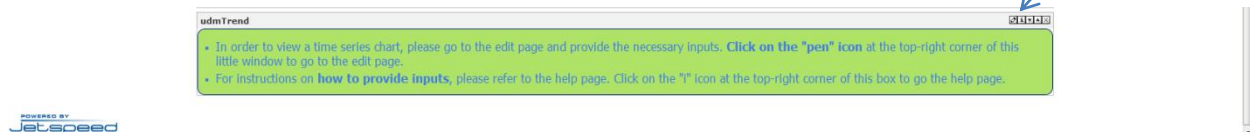
3. This search produces the two udm chart options.



4. Click the “Add” option under the type of chart you want to produce. In the example below, a chart for udmTrend was selected.



5. An instructions portlet like the one in the image below will be produced. Click the **i** button shown in the image below



6. This will take you to the edit page. On this page you can add data, adjust the chart height and the amount of data displayed, adjust the gridline for the x-axis and y-axis, and change the title for the chart, x-axis, and y-axis.

7. To add data to the chart click “Add points.”

4.3.2 Adding Points:

Single Point

8. A pop up window will appear.

9. To find a specific set of points to add to the chart, enter information about the parameters such as “Site” and “Point.Description.” (Example shown in the image below) Then, click on “Search.”

Select	Project	Site	Point.Description	Point.UOM	Point.Forecast
<input type="checkbox"/>	APC Dual Hex Unit Tests	LBNL_Bldg1275	APC-Chilled-Thermal-Rate	kW	false
<input type="checkbox"/>	APC Dual Hex Unit Tests	LBNL_Bldg1275	APC-Chilled-Water-Flow	GPM	false
<input type="checkbox"/>	APC Dual Hex Unit Tests	LBNL_Bldg1275	APC-Chilled-Water-Supply-Temp	F	false
<input type="checkbox"/>	APC Dual Hex Unit Tests	LBNL_Bldg1275	APC-Chilled-Water-Return-Temp	F	false

10. To select the set of points you want on the chart, check the box next to the point set as seen in the image below.

Jetspeed

Public Space >> udmTrendSearch >> Space Home

udmTrendSearch

Search for Points by Filtering against Point Details.

Point Search

Project: Site: 1275

Point.ID: Point.Description: chill

Point.UOM: Point.Forecast:

Point Selection

Select	Project	Site	Point.Description	Point.UOM	Point.Forecast
<input checked="" type="checkbox"/>	APC Dual Hex Unit Tests	LBNS_Bldg1275	APC-Chilled-Thermal-Rate	kW	false
<input type="checkbox"/>	APC Dual Hex Unit Tests	LBNS_Bldg1275	APC-Chilled-Water-Flow	GPM	false
<input type="checkbox"/>	APC Dual Hex Unit Tests	LBNS_Bldg1275	APC-Chilled-Water-Supply-Temp	F	false
<input type="checkbox"/>	APC Dual Hex Unit Tests	LBNS_Bldg1275	APC-Chilled-Water-Return-Temp	F	false

11. Then, click “Add Checked Points” to add the point set to the chart. Finally, click “Close Window” to close this window.

12. Now in the main window, check the box of the set of points that you want displayed on the chart. As seen in the image below the set of points for “APC Dual Hex Unit Tests” was selected.

udmTrend

Configure your portlet by choosing different parameters.

Look and Feel

Chart Height (pixels): 400 Note: Your screen height is 1000 pixels. Chart Title: Sample

X-Axis Label: Time Y-Axis Label: Value

X-Axis Gridline: ☒ Y-Axis Gridline: ☒

Time

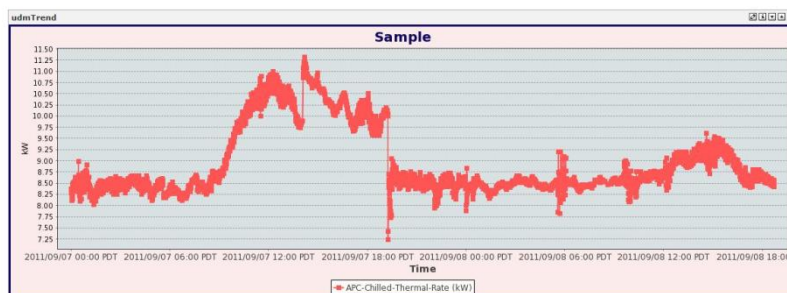
Start Time: Y End Time: now

Point Selection

Select	Project	Site	Point.Description	Point.UOM	Point.Forecast
<input checked="" type="checkbox"/>	APC Dual Hex Unit Tests	LBNS_Bldg1275	APC-Chilled-Thermal-Rate	kW	false

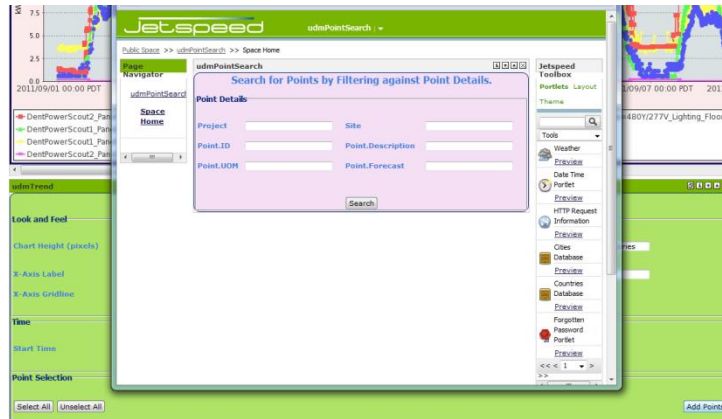
13. Now, click **Apply** to create a chart containing this set of points.

14. Now the chart of points has been created.



Multiple points

8. A pop up window will appear.



9. To find specific point sets to add to the chart, enter information about the parameters such as “Site” and “Point.Description.” (Example shown in the image below) Then, click on “Search.” (To view all possible point sets, click “Search” without enter any parameters such as “Site” or “Point.Description”)



10. To select the sets of points you want on the chart, check the boxes next to the point sets as shown in the image below.

Point Search

Project: Site: 1275

Point.ID: Point.Description: chill

Point.UOM: Point.Forecast:

Point Selection

Select	Project	Site	Point.Description	Point.UOM	Point.Forecast
<input checked="" type="checkbox"/>	APC Dual Hex Unit Tests	LBNL_Bldg1275	APC-Chilled-Thermal-Rate	kW	false
<input checked="" type="checkbox"/>	APC Dual Hex Unit Tests	LBNL_Bldg1275	APC-Chilled-Water-Flow	GPM	false
<input checked="" type="checkbox"/>	APC Dual Hex Unit Tests	LBNL_Bldg1275	APC-Chilled-Water-Supply-Temp	F	false
<input type="checkbox"/>	APC Dual Hex Unit Tests	LBNL_Bldg1275	APC-Chilled-Water-Return-Temp	F	false

11. Then, click “Add Checked Points” to add the point sets to the chart. Finally, click “Close Window” to close this window.

12. Now in the main window, check the boxes for the sets of points that you want displayed on the chart.

Look and Feel

Chart Height (pixels): 400 Chart Title: Sample

X-Axis Label: Time Y-Axis Label: Value

X-Axis Gridline: ☒ Y-Axis Gridline: ☒

Time

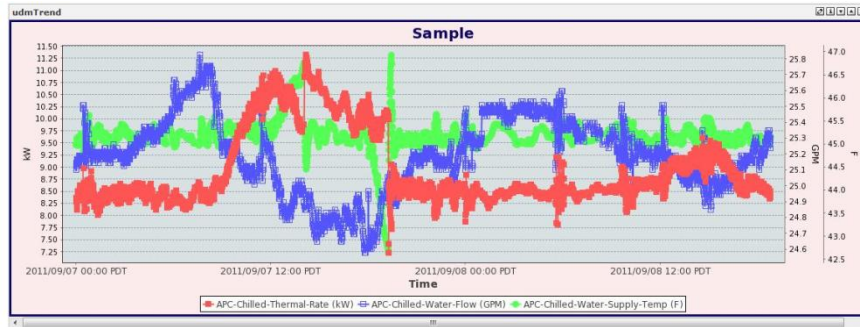
Start Time: y End Time: now

Point Selection

Select	Project	Site	Point.Description	Point.UOM	Point.Forecast
<input checked="" type="checkbox"/>	APC Dual Hex Unit Tests	LBNL_Bldg1275	APC-Chilled-Thermal-Rate	kW	false
<input checked="" type="checkbox"/>	APC Dual Hex Unit Tests	LBNL_Bldg1275	APC-Chilled-Water-Flow	GPM	false
<input checked="" type="checkbox"/>	APC Dual Hex Unit Tests	LBNL_Bldg1275	APC-Chilled-Water-Supply-Temp	F	false

13. Now, click [Apply](#) to create a chart containing these sets of points.

14. Now the chart of points has been created.



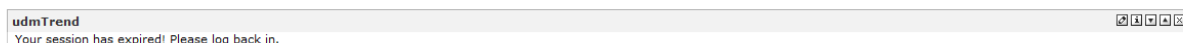
4.3.3 Common Errors

Attempt to display the same point set multiple times on the same chart:

Select	Project	Site	Point Description	Point UOM	Point Forecast
<input checked="" type="checkbox"/>	APC Dual Hex Unit Tests	LDNS_48dgl175	APC-Chilled-Thermal-Rate	kW	None
<input checked="" type="checkbox"/>	APC Dual Hex Unit Tests	LDNS_48dgl175	APC-Chilled-Thermal-Rate	kW	None
<input checked="" type="checkbox"/>	APC Dual Hex Unit Tests	LDNS_48dgl175	APC-Chilled-Water-Flow	GPM	None
<input checked="" type="checkbox"/>	APC Dual Hex Unit Tests	LDNS_48dgl175	APC-Chilled-Water-Supply-Temp	F	None

(Duplicate point set: “APC Dual Hex Unit Tests”| Point.UOM-kW)

When you attempt to create a chart using duplicate point sets, the result of clicking “Apply” will not be a chart but rather an error message.



5 Download Data from UDM

The udmDataDump.zip package needs to be used for this. Download the package, unzip it and modify the script file to reflect your site specific configurations and your need. The sample input file has been provided as a guideline, the content of the input file will be specific to the users need. The command in the script is as follows.

```
java -jar udmDataDump.jar hostname username password InputPointFilename start_time end_time
start_BoundaryType end_BoundaryType Outputvalue_file wantCanonicalName outputTimeFormat
printOnConsole output_type interpolation_type interpolation_time_interval
```

The explanation of all the inputs are provided below.

Argument	Meaning
InputPointFilename	The file where the point details are stored. These details, together, should uniquely identify each point in the database. If just the filename is specified, instead of the absolute path, the file is assumed to be in the same directory as the batch file
start_time	Start point of the time range
end_time	End point of the time range
start_BoundaryType	It should any one of ">=" and ">" - it is used to define the boundary of the time range.
end_BoundaryType	It should any one of "<=" and "<" - it is used to define the boundary of the time range.
Outputvalue_file	The file where the final output is written. If just the filename is specified, instead of the the absolute path, the file is assumed to be in the same directory
wantCanonicalName	In order to uniquely identify a point in the database, a combination of Project, Site, Description and DataSourceType is required. Therefore, if this boolean flag is true, the column header in the output file would read Project--Site--Description--DataSourceType. However, often, just the description turns out to be sufficient to uniquely identify the point in the output file. Hence, for the ease of readability of the output file

Argument	Meaning
	one can turn this flag to false.
outputTimeFormat	The format in which the user wants the timestamp. An example is "yyyy/MM/dd HH:mm:ss"
printOnConsole	This boolean flag indicates whether the values should be printed on the console in addition to being written to the file
output_type	"raw" or "interpolated"
interpolation_type	"point_based" or "fixed_interval" - In case of "point_based" interpolation, the timestamps to which the interpolation is done are essentially timestamps of a specific point. To be specific, it is the first point in the list of input points. In case of fixed_interval, the interpolation timestamp is calculated as $(start_time + n * interpolation_time_interval)$
interpolation_time_interval	0 if interpolation_type is "point_based" and the value in seconds if interpolation_type is "fixed_interval"

6 The Scripts Package

In order to aid system setup on the linux OS, a few scripts have been created. They are available in the package script.zip.

6.1 The *StartStop scripts

The *StartStop.sh scripts are samples, with installation specific directory names. So, please check the content of the script, specifically the directory names, before using them. These scripts can be used to start or stop the respective program. For example, the following command will start the Modbus ProtocolCommunicator and run it in the background.

```
./udmModbusStartStop.sh start
```

To stop the process, issue the following command.

```
./udmModbusStartStop.sh stop
```

NOTE: they need to be run under a non-root user.

6.2 Automated startup at system startup (linux)

These *StartStop scripts (or links to them) can be placed in the /etc/init.d directory for the processes to start at system start-up.

6.3 create_udm_home.sh

The value of the environment variable UDM_HOME is set in this script. This script is, in turn, called by all the udm*StartStop scripts. It serves the following purposes:

- 1) It ensures that the value of the environment variable UDM_HOME is set when the *StartStop scripts are being run at system startup.
- 2) If the UDM_HOME directory is ever changed, it needs to be updated only at one place, and that is, in the create_udm_home.sh script.

6.4 `create_user.sh`

This script can be used to create the non-root user under which the *StartStop scripts can run.

6.5 `backup.sh`

This script can be run to backup the UDM database.

7 PostgreSQL Installation

On the Windows OS, installation of PostgreSQL is very straight forward. For, linux, a script called `postgresql_installation.sh` has been created. So, for linux OS:

1. Download the package `script.zip` and unzip it
2. Run `postgresql_installation.sh`

Command: `./postgresql_installation $1 $2`

\$1: the directory which will contain the `udm` and `udm packages` directory

\$2: the group for the user, who is created during the script, that will control the postgresql database, `postgres`

Example: `./postgresql_installation /data/software users`

8 UDM Initial Setup

8.1 Unified Data Manager Setup for a Database Host

1. Download the UDM Distribution Packages into a directory that will contain all the udm zip directories
2. Next unzip the zip directory, udmDatabaseSetup.zip.
3. Once unzipped, go into the udmDatabaseSetup directory and read the file, readme.txt.
4. Follow the directions given in that text file.
5. Next, copy all of the udm zip directories, except udmDatabaseSetup.zip, into the directory, udm, that was created in step 4.
 - a. command: `cp *.zip /path/to/udm` (example: `cp *.zip /data/software/udm`)
6. Unzip all of the zip directories in the udm directory
7. Inside an unzipped directory
 - a. Run the command `dos2unix` on the .sh file
 - b. Run the command `chmod ugo+x` on the .sh file
 - c. Modify the content of the .sh file to suit the site specific configuration
 - d. Run the .sh file
8. If more than one user needs to work on an udm application, run the command `chmod -R g+w` on that application's unzipped directory

8.2 Unified Data Manager Setup for a Non-Database Host

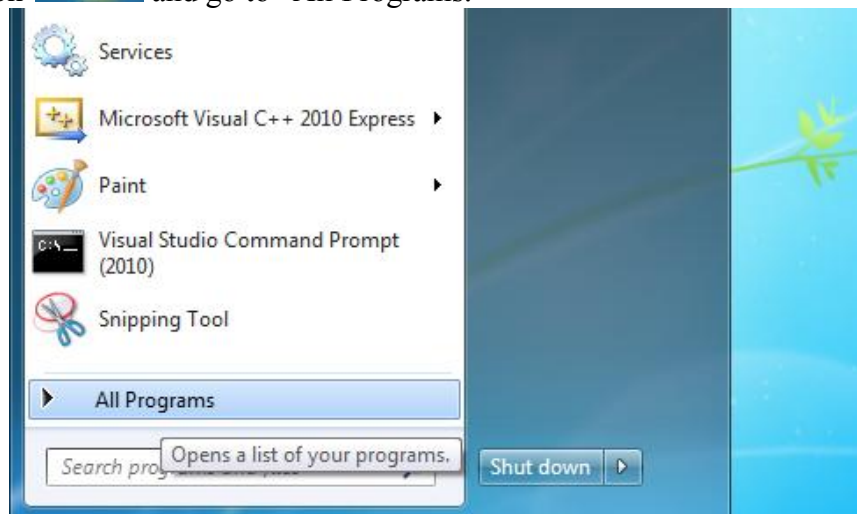
UDM setup

1. Download the UDM Distribution Packages into a directory that will contain all the udm zip directories
2. Next unzip the zip directory, udmClientSetup.zip.
3. Once unzipped, go into the udmClientSetup directory and read the file, readme.txt.
4. Follow the directions given in that text file.
5. Next, copy all of the udm zip directories, except udmDatabaseSetup.zip, into the directory, udm, that was created in step 4.
 - a. command: `cp *.zip /path/to/udm` (example: `cp *.zip /data/software/udm`)
6. Unzip all of the zip directories in the udm directory
7. Inside an unzipped directory

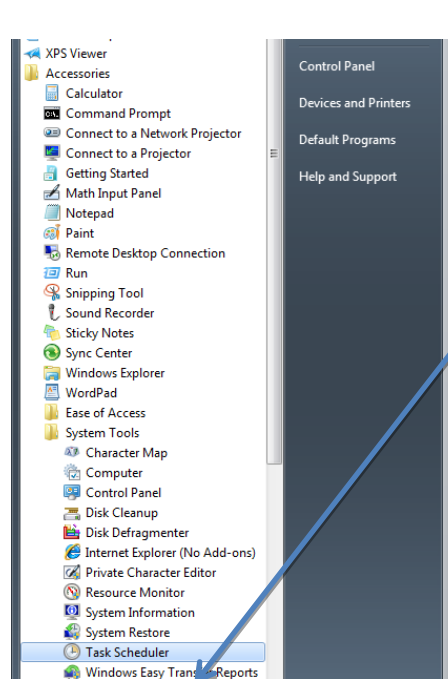
- b. Run the command *dos2unix* on the .sh file
 - b. Run the command *chmod ugo+x* on the .sh file
 - c. Modify the content of the .sh file to suit the site specific configuration
 - d. Run the .sh file
8. If more than one user needs to work on an udm application, run the command *chmod -R g+w* on that application's unzipped directory

8.3 How to Setup UDM Programs to Run at Windows StartUp

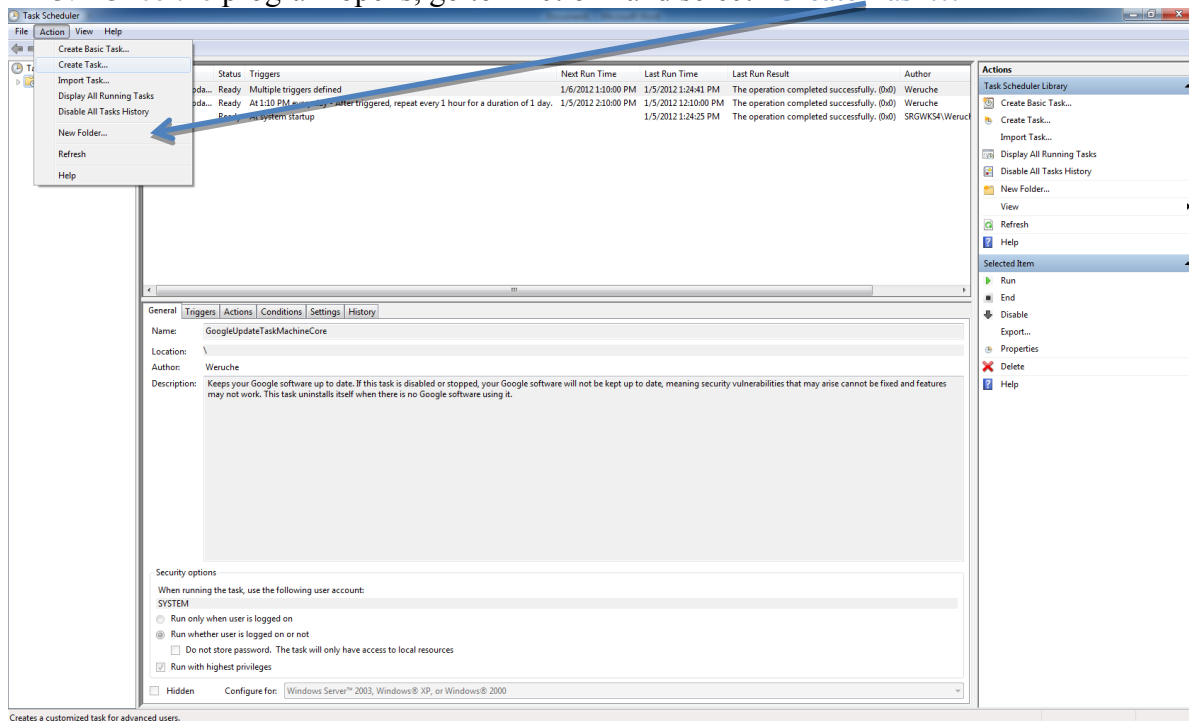
1. Click on  and go to "All Programs."



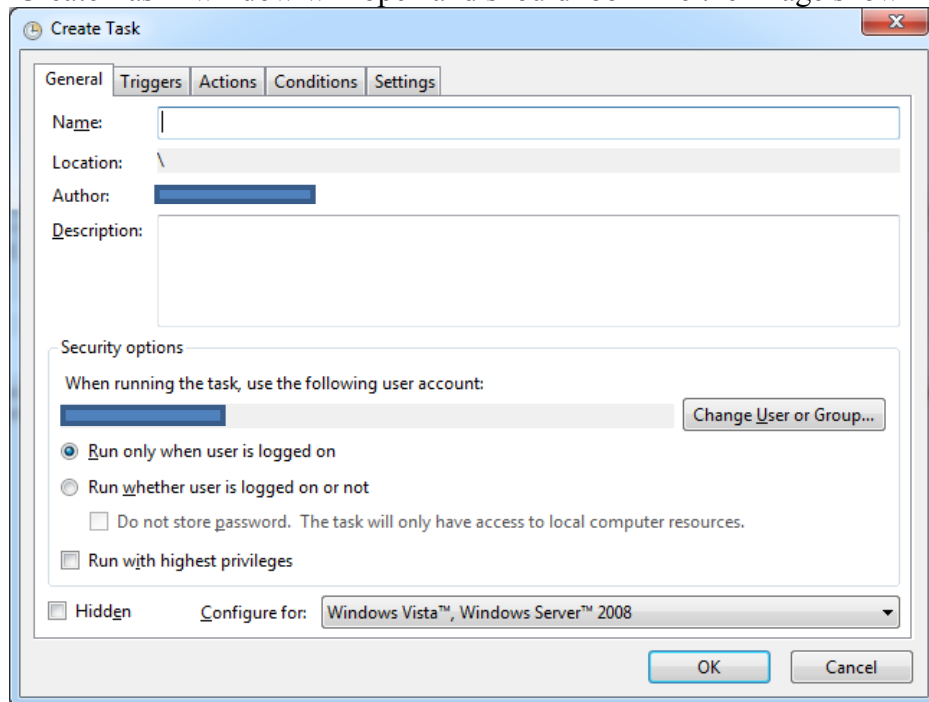
2. Next, click on “Accessories” and go to the folder “System Tools.” Inside that folder will be the program “Task Scheduler.” As shown in the image below, select “Task Scheduler.”



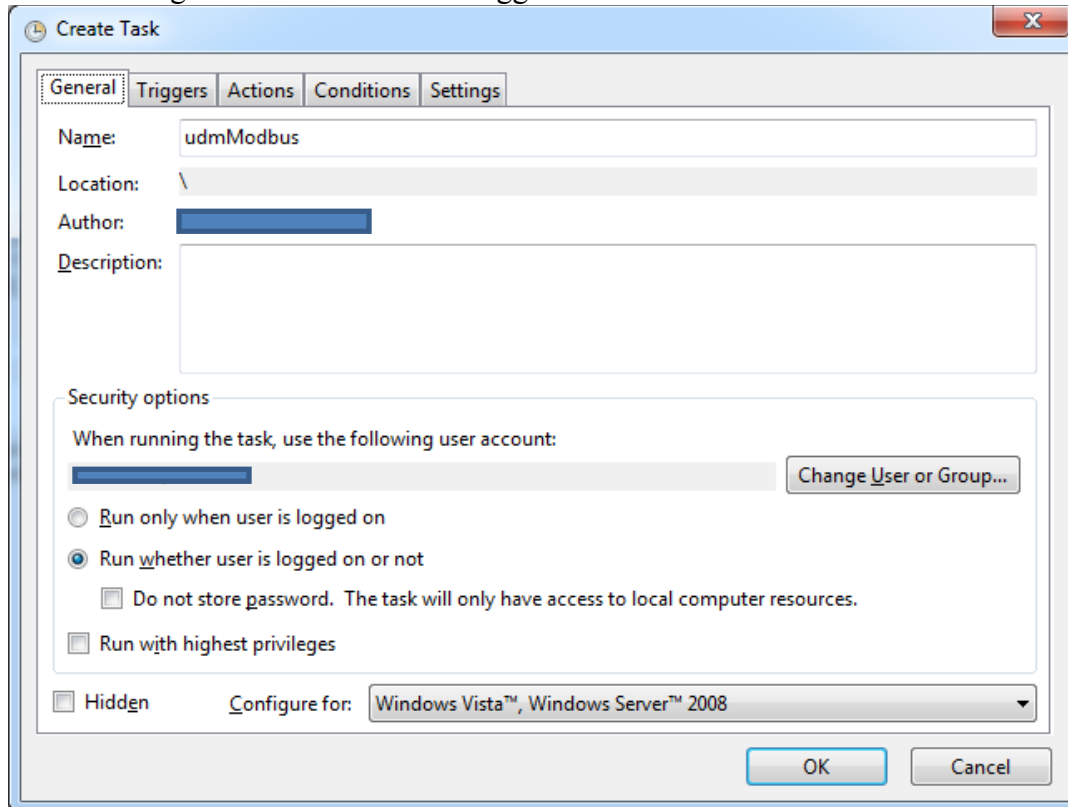
3. Once the program opens, go to “Action” and select “Create Task...”



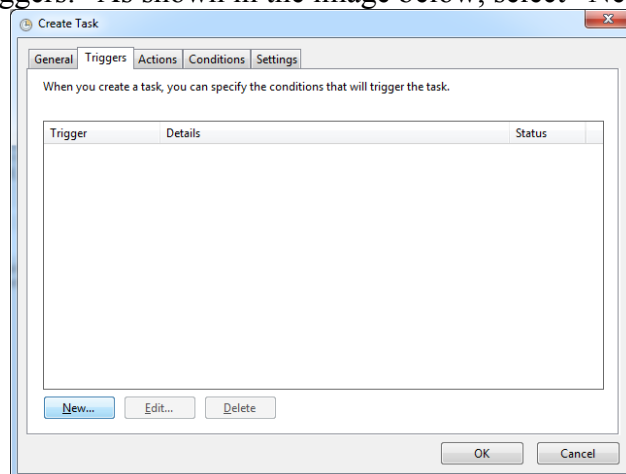
4. The “Create Task” window will open and should look like the image shown below.



5. In this tutorial, we will be using the udm application, udmModbus, as the program we are setting up to start at system startup. Consequently, we named our schedule task udmModbus. Under “Security options” configure your task to run according to your needs. In the image shown below, the only change we have made to “Security options” was selecting “Run whether user is logged on or not”



6. Next, go to “Triggers.” As shown in the image below, select “New...”



7. This will open a new window called “New Trigger.” Go to the scroll down menu, “Begin the task” and select the option “At startup.” Then, press “OK” to exit the window.

New Trigger

Begin the task: At startup

Settings

No additional settings required.

Advanced settings

☐ Delay task for: 15 minutes

☐ Repeat task every: 1 hour for a duration of: 1 day

☐ Stop all running tasks at end of repetition duration

☐ Stop task if it runs longer than: 3 days

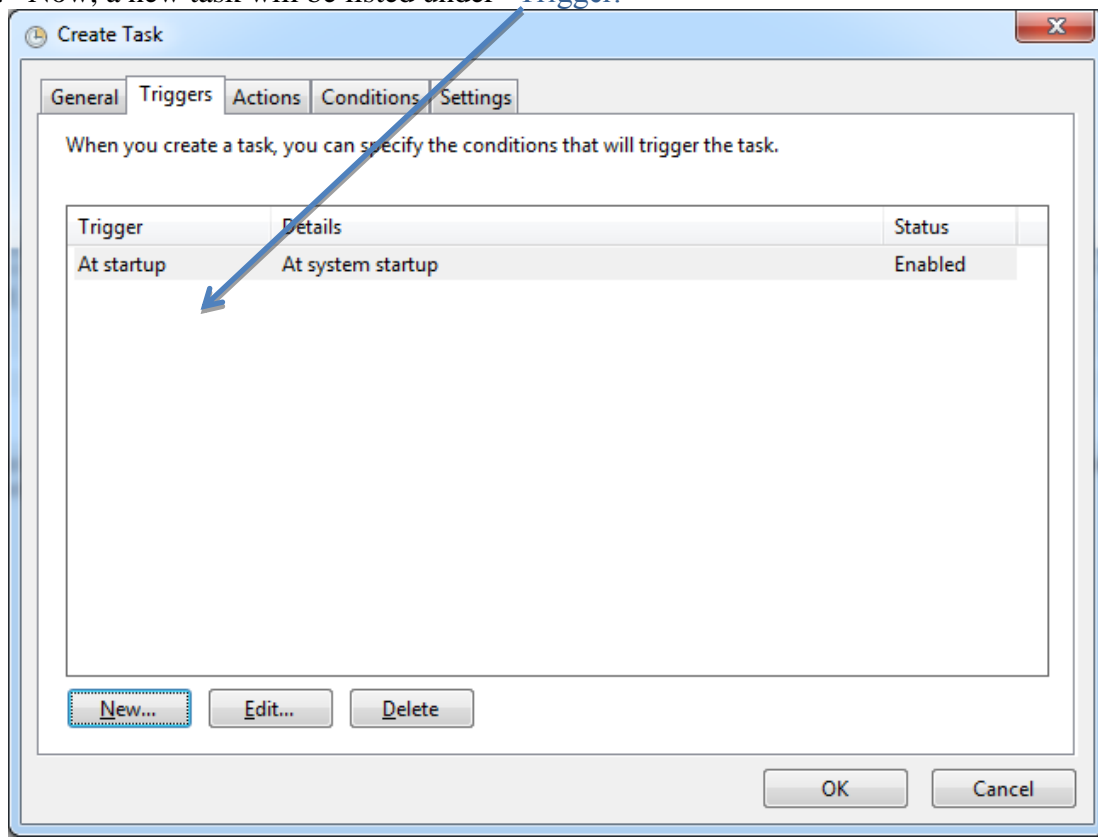
☐ Activate: 1/ 5/2012 1:49:24 PM ☐ Synchronize across time zones

☐ Expire: 1/ 5/2013 1:49:24 PM ☐ Synchronize across time zones

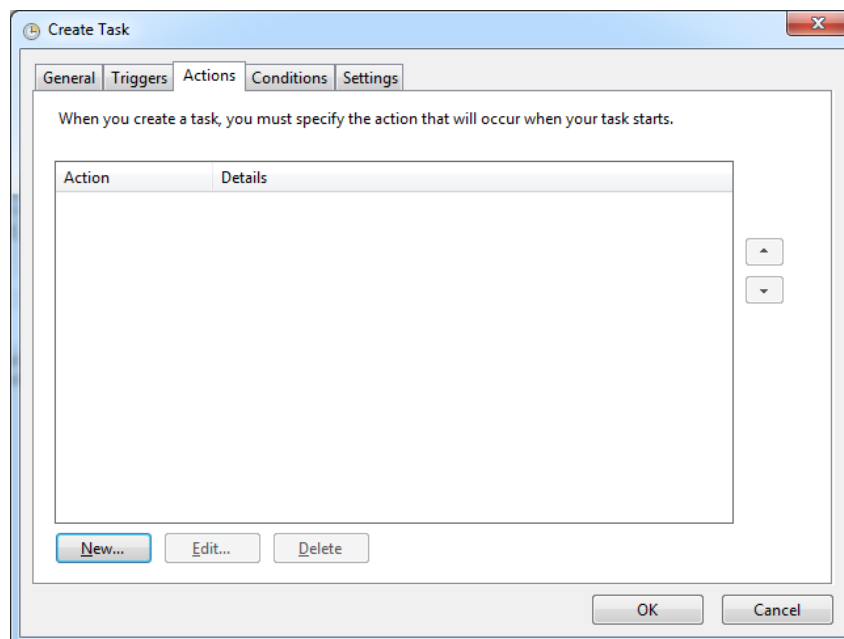
☒ Enabled

OK **Cancel**

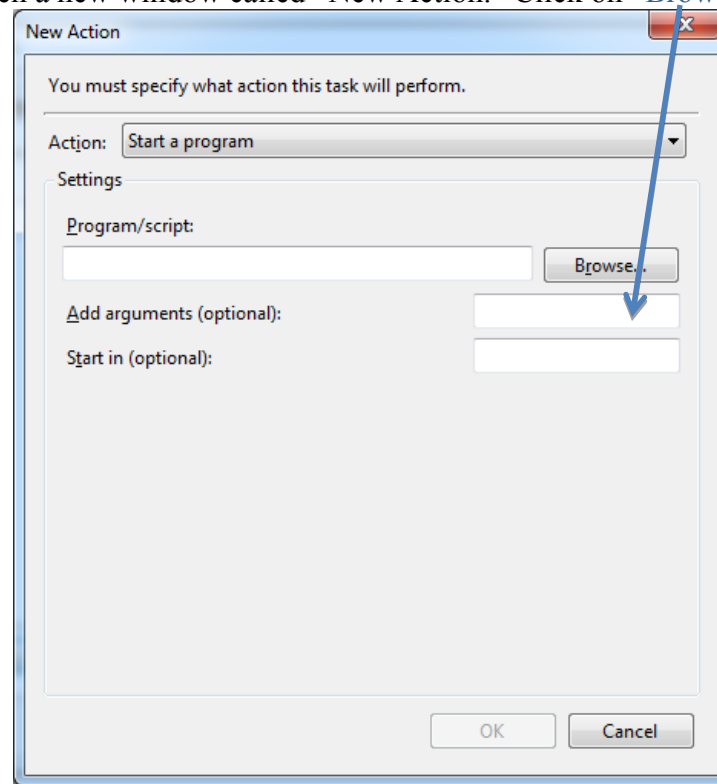
8. Now, a new task will be listed under “Trigger.”



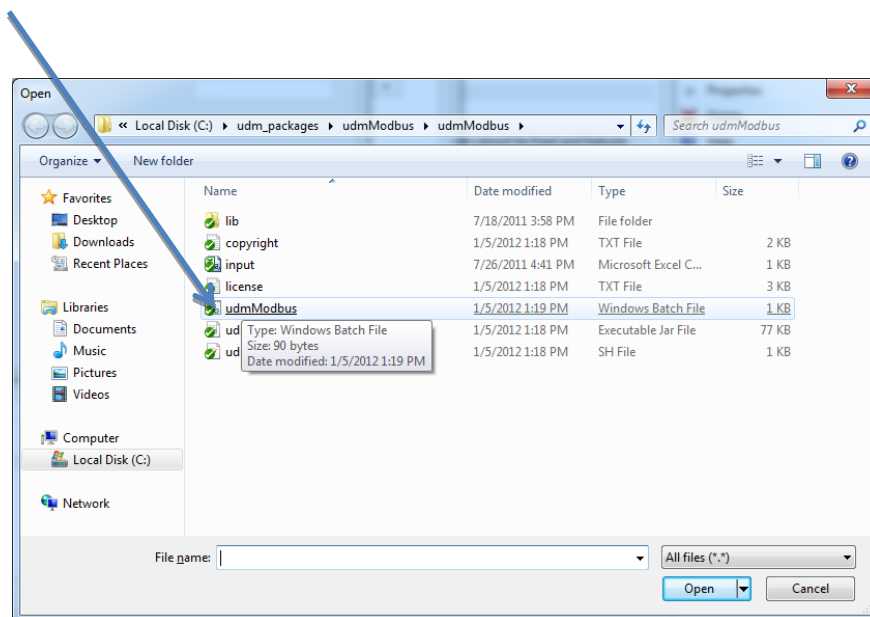
9. Next, go to “Actions.” As shown in the image below, select “New...”



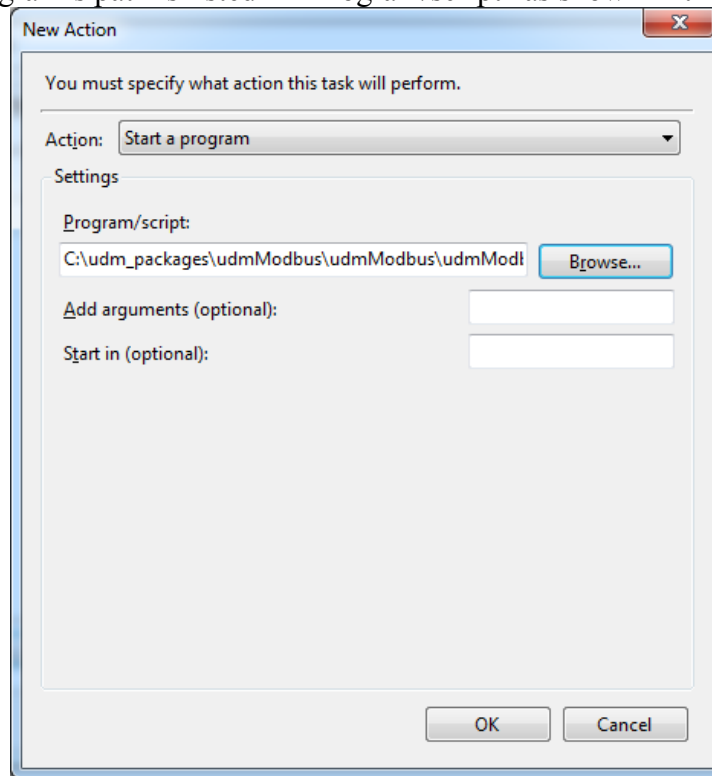
10. This will open a new window called “New Action.” Click on “Browse...”



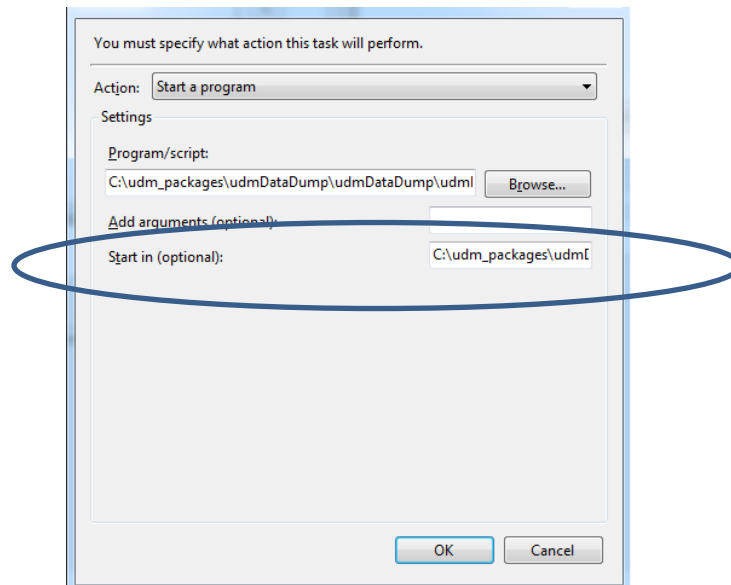
11. This will open a new window. Go to the place where your udm programs are stored and select the .bat file that launches the udm program. In this tutorial, we will be using the udm application, udmModbus. Consequently, we choose the batch file called udmModbus.



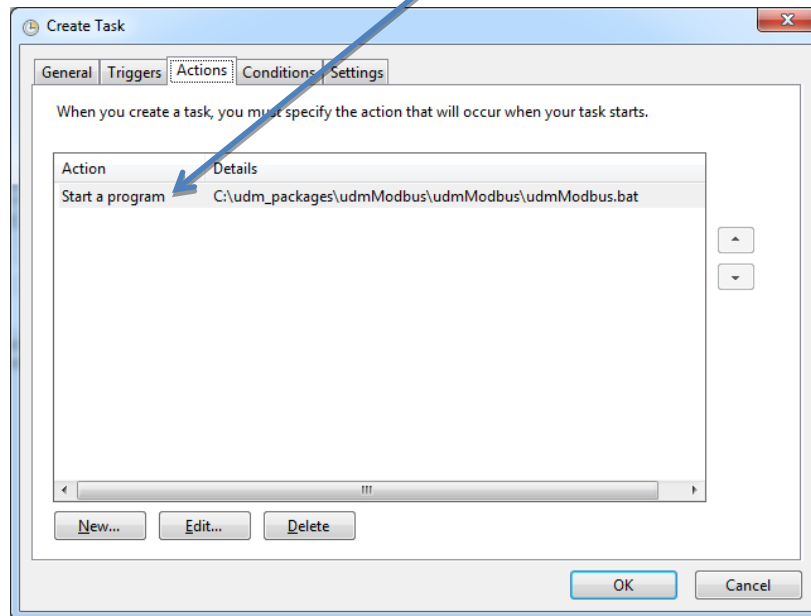
12. Now the program's path is listed in "Program/script" as shown in the image below.



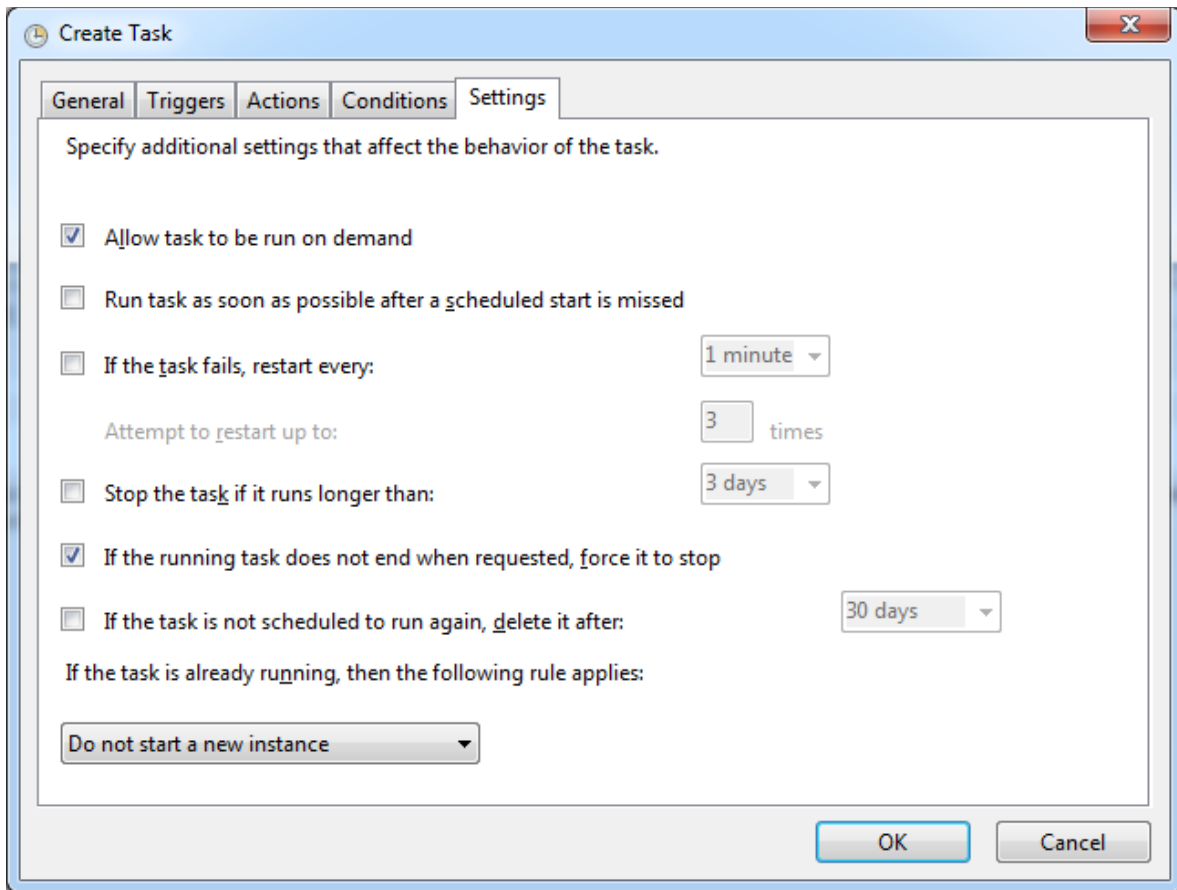
13. Next, copy the path listed in "Program/script." Next, paste the path in "Start in (optional):" and delete the part of the path that includes the batch file name.
- Example: if the path in "Program/script" is `C:\udm\udmModbus\udmModbus.bat` the path in "Start in (optional):" would be `C:\udm\udmModbus\`
- Then, press "OK" to exit the window.



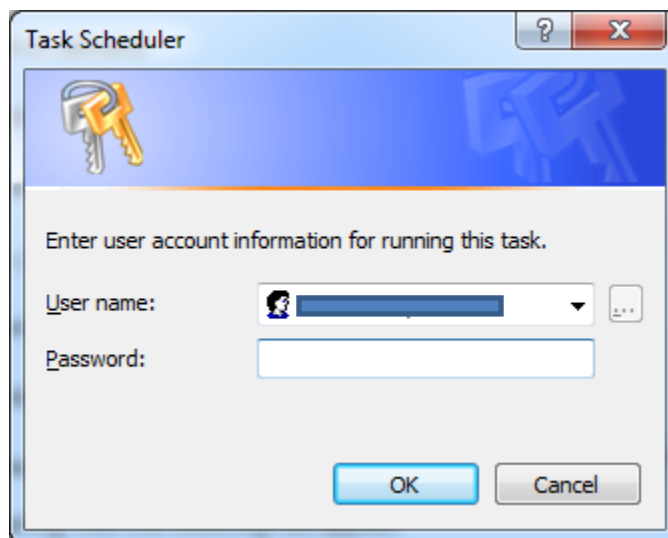
14. Now, a new program will be listed under “Action.”



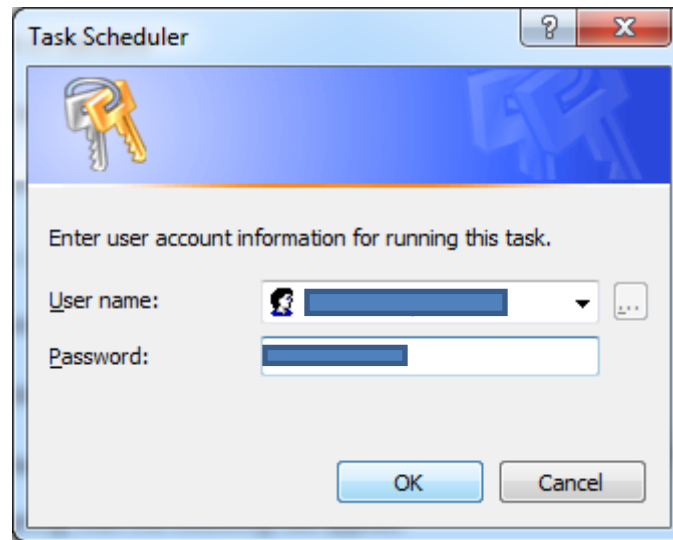
15. Next, go to “Settings.” Select the options shown in the window below.



16. Then, press “OK.” The following window will open:

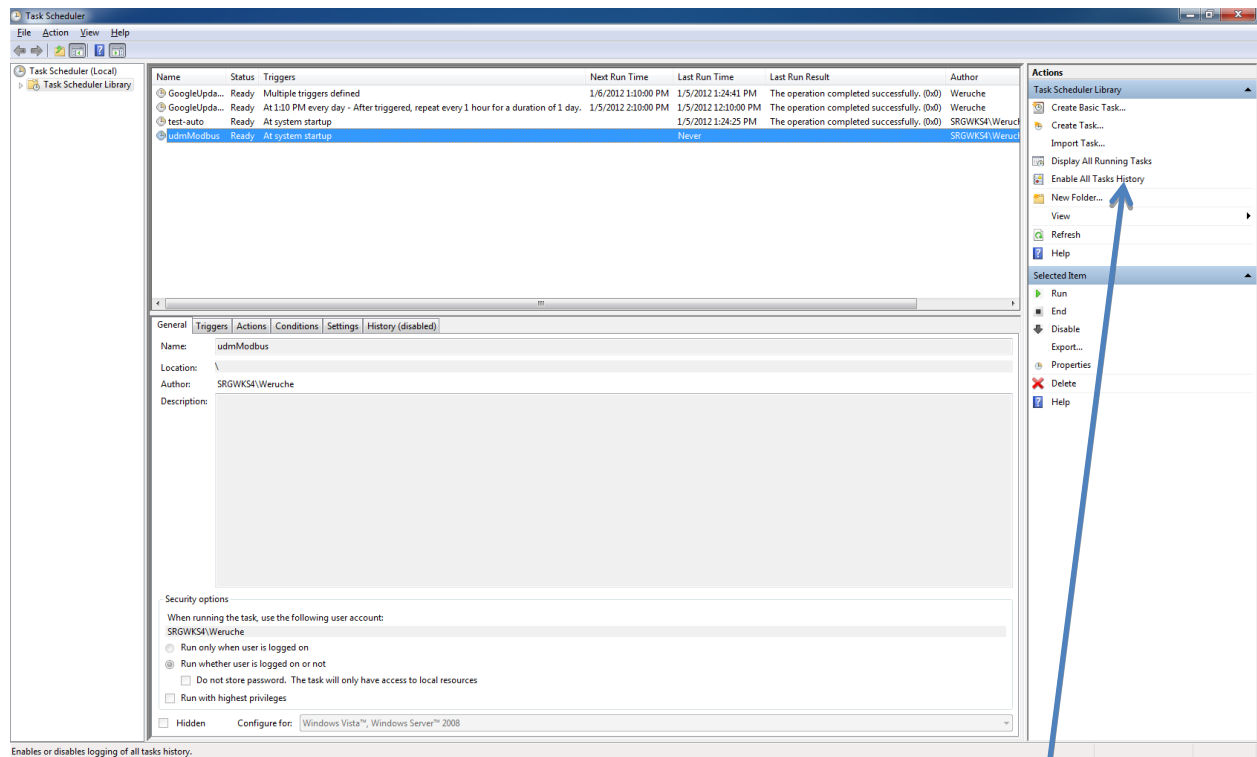


17. Enter your password and press “OK.”



18. This newly created task will now be listed in the task scheduler as shown in the image below.





19. If you have not already enabled task history, click on “Enable All Tasks History” to enable task history. Once enabled, it will show “Disable All Tasks History.”

9 SysAdmin Tool

In order to create, edit or delete Points in the UDM database or to import the list of existing Points from the database, one needs to use this tool. This tool is available in the `udmSysAdmin.zip` package. This tool can be run locally on the UDM server or remotely from a machine already configured as a UDM client. The command looks like the following.

```
java -jar udmSysAdmin.jar hostname username password %1 inputFilename
outputFilename
```

Following is an explanation of the different arguments.

hostname = hostname, IP address or FQDN of the database server

username = the username to access the database

password = password for this username

This script takes ONLY one input. The input is the type of action to be performed. Valid inputs are: `createPoint`, `editPoint`, `deletePoint` and `importPoint`. These inputs are case-insensitive.

- `createPoint` is used when the intention is to create Points in the database – requires the ID column to be ABSENT in the input csv file
- `editPoint` is used when the intention is to edit Points in the database – requires the ID column to be PRESENT in the input csv file
- `deletePoint` is used when the intention is to delete Points from the database – requires the ID column to be PRESENT in the input csv file
- `importPoint` is used when the intention is to import the details about all the Points in the database

inputFilename = csv filename. This file is used when `createPoint` and `deletePoint` options are chosen.

Information contained in this file is used for creating or deleting Points.

outputFilename = csv filename. This file is used when `importPoint` option is chosen. All the Point information is output into this file.

10 Jetspeed Installation

This document will discuss the following items:

- i. How to install Jetspeed in a Linux operating system
- ii. How to configure Jetspeed to use a non-default database, postgresql
- iii. How to run Jetspeed as a service

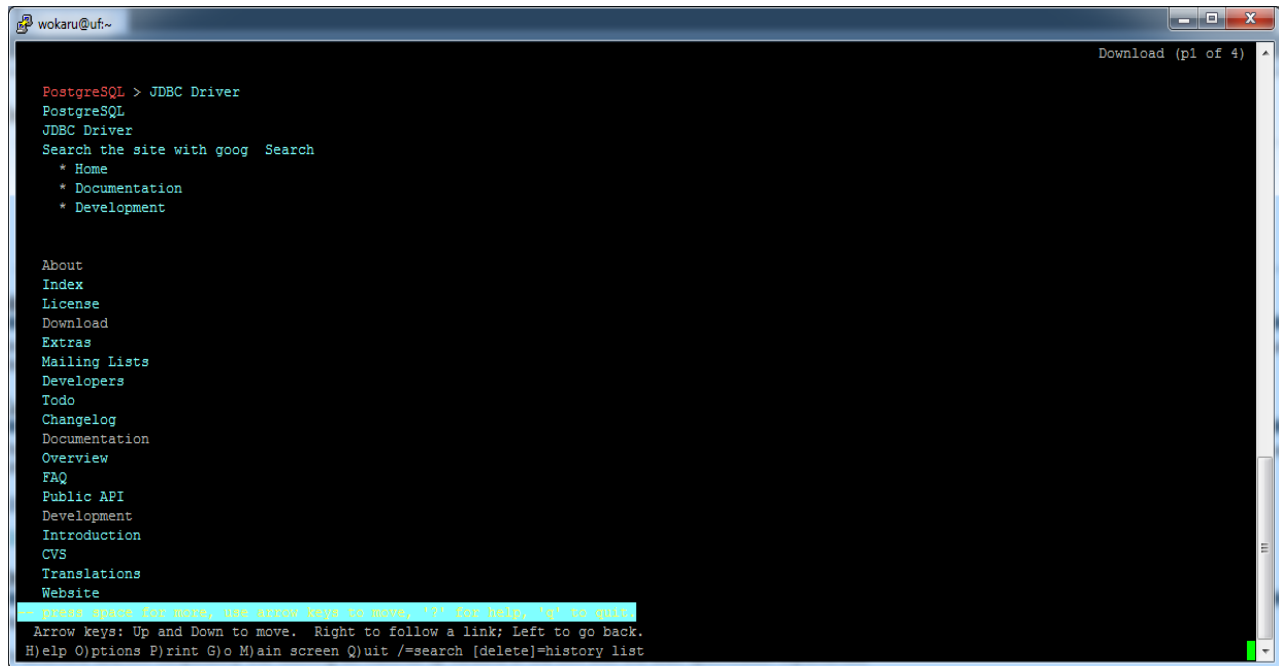
10.1 Download the JDBC driver

1. First, check which version of java is running on your system using the command,
java -version

```
wokaru@uf:~  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$  
[wokaru@uf ~]$ java -version  
java version "1.6.0_17"  
OpenJDK Runtime Environment (IcedTea6 1.7.10) (rhel-1.21.b17.el5-x86_64)  
OpenJDK 64-Bit Server VM (build 14.0-b16, mixed mode)  
[wokaru@uf ~]$
```

2. In order to run Jetspeed with a postgresql database, a jdbc driver must also be installed. To download this driver, go the website <http://jdbc.postgresql.org/download.html>. In the command line type in the following

command, *lynx http://jdbc.postgresql.org/download.html* and press enter. Then, you will be taken to the webpage as seen below:



3. Next, scroll down using the down arrow on the keyboard. Stop when you reach the Downloads section. Go the subsection, Current Version, and select which version to download. "If you are using the 1.6 JVM, then you should use the JDBC4 version" (jdbc.postgresql.org). Since, the JVM in this document is 1.6, JDBC4 version was selected.

```
wokaru@uf-  
  
Private API  
2  
Download  
* About  
  Current Version  
  Other Versions  
  Development Versions  
  Supported Versions  
  Archived Versions  
About  
Binary JAR file downloads of the JDBC driver are available here. Because Java is platform neutral, it is a simple process of just downloading the appropriate JAR file and dropping it into your classpath. Source versions are also available here for recent driver versions. Prior to the 8.0 release the JDBC driver was distributed with the server source code.  
Current Version  
This is the current version of the driver. Unless you have unusual requirements (running old applications or JVMs), this is the driver you should be using. It supports PostgreSQL 7.2 or newer and requires a 1.4 or newer JVM. It contains support for SSL and the javax.sql package. It comes in two flavors, JDBC3 and JDBC4. If you are using the 1.6 JVM, then you should use the JDBC4 version.  
JDBC3 PostgreSQL Driver, Version 9.0-801  
JDBC4 PostgreSQL Driver, Version 9.0-801  
Other Versions  
Download Options (Lynx Version 2.8.5rel.1), help  
Arrow keys: Up and Down to move. Right to follow a link; Left to go back.  
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

4. After pressing enter, you will be directed to the Download options page.

```
wokaru@uf-  
Download Options (Lynx Version 2.8.5rel.1), help  
Downloaded link: http://jdbc.postgresql.org/download/postgresql-9.0-801.jdbc4.jar  
Suggested file name: postgresql-9.0-801.jdbc4.jar  
Standard download options:  
  Save to disk  
Local additions:  
  View with less  
Command: Use arrow keys to move, '?' for help, 'q' to quit, 'b' to go back  
Arrow keys: Up and Down to move. Right to follow a link; Left to go back.  
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

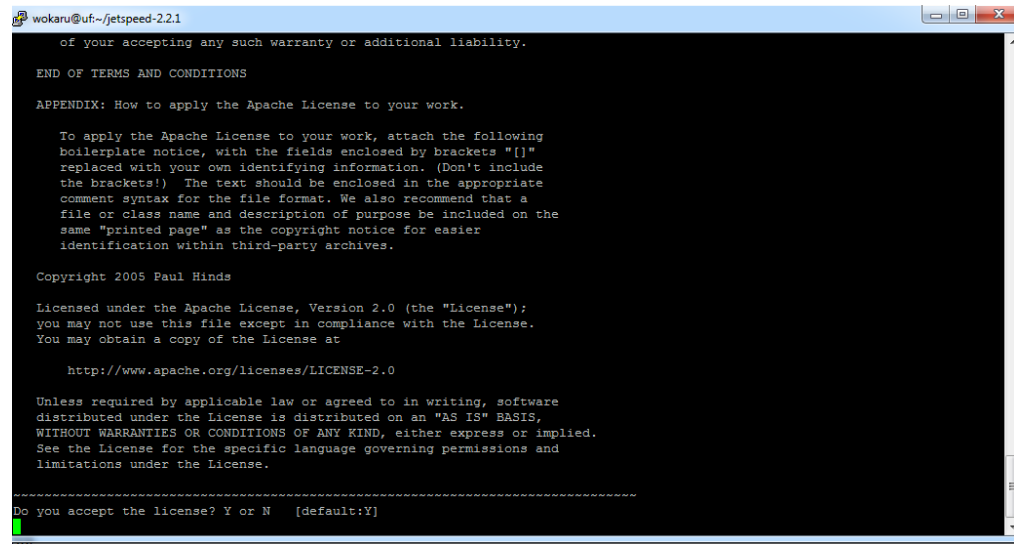
5. As seen in the image below, select *Save to disk*

Example: `wget http://www.apache.org/dyn/closer.cgi/portals/jetspeed-2/binaries/jetspeed-installer-2.2.1.jar`

2. To run the installer, using the `java -jar` command:

Example: `java -jar jetspeed-installer-2.2.1.jar`

3. As shown below, the following page appears in the command prompt once the installer has been started.

A screenshot of a terminal window titled 'wokaru@ufi:~/jetspeed-2.2.1'. The window displays the Apache License, Version 2.0, in a monospaced font. The text includes the license terms, a copyright notice for 2005 Paul Hinds, and a prompt asking 'Do you accept the license? Y or N [default:Y]'. The terminal has a black background with white text, and the window has standard Linux window controls (minimize, maximize, close) in the top right corner.

```
wokaru@ufi:~/jetspeed-2.2.1
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright 2005 Paul Hinds

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

~~~~~
Do you accept the license? Y or N  [default:Y]
```

4. To continue with the Jetspeed installation, type “Y” into the command prompt and press enter.
5. Next, you will be asked which operation you want the installer to perform. As seen in the image below, enter 1 to install a Jetspeed portal

```
-----
Please select the operation the installer is to perform.
Operation
Enter a number
1) Install Jetspeed Portal
2) Export Jetspeed Portal database
3) Initialize Jetspeed Portal database
4) Import Jetspeed Portal database
1

-----
Installation location
-----
Select an installation directory [default:/home/wokaru/Jetspeed-2.2.1]

```

6. Then,

```
6) Oracle 9
7) Oracle 10 or higher
8) SapDB
9) Manual database setup
5

-----
PostgreSQL Database connection parameters
-----

Please fill in the fields below for connecting to the PostgreSQL database.
Database User Name [default:]
jetspeed

Database Password [default:]
postgres

JDBC Connection String [default:jdbc:postgresql://localhost:5432/<dbname>]
jdbc

```

7.

```
6) Oracle 9
7) Oracle 10 or higher
8) SapDB
9) Manual database setup
5

-----
PostgreSQL Database connection parameters
-----

Please fill in the fields below for connecting to the PostgreSQL database.
Database User Name  [default:]
jetspeed

Database Password  [default:]
postgres

JDBC Connection String  [default:jdbc:postgresql://localhost:5432/<dbname>]
jdbc
```

10.3 How to Use the Jetspeed Daemon

Description

The script, tomcat, allows Jetspeed to run as a daemon¹.

Platform: Red Hat 4.1.2-50 (CentOS release 5.6)

Installation

Copy the script, tomcat, from the Jetspeed directory into where the system's init scripts are stored (In this tutorial, this directory will be referred to as /etc/init.d). To verify that tomcat has been copied into the /etc/init.d directory run the command `ls` in this directory. Once it has been verified that the script has been successfully copied into the /etc/init.d directory, run the command `chkconfig --add tomcat`. To verify that udmModbus has been added to the run levels, run the command `chkconfig --list`.

Operation

To start, stop or check the status of Tomcat in any directory use the commands:

```
sudo /etc/init.d/tomcat start
```

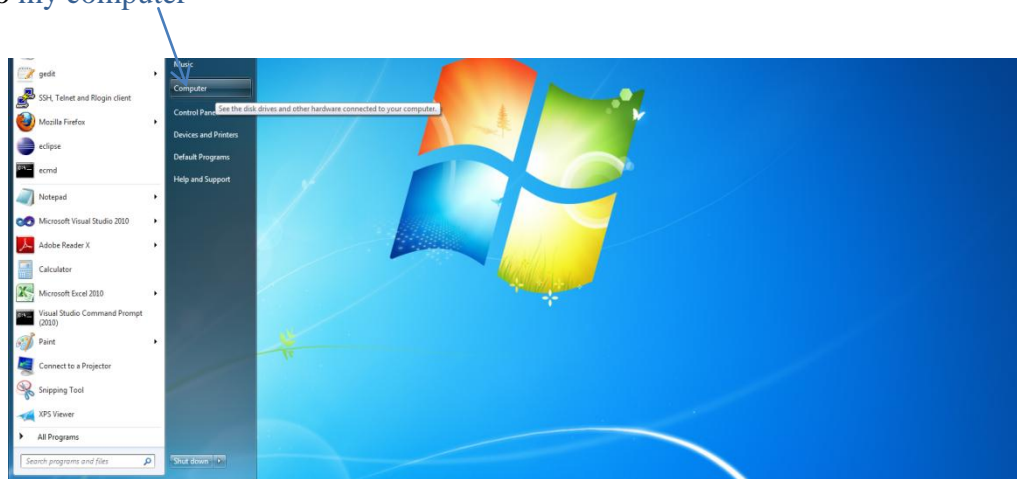
```
sudo /etc/init.d/tomcat stop
```

¹ Daemon: A background process that handles requests for services such as print spooling and is dormant when not

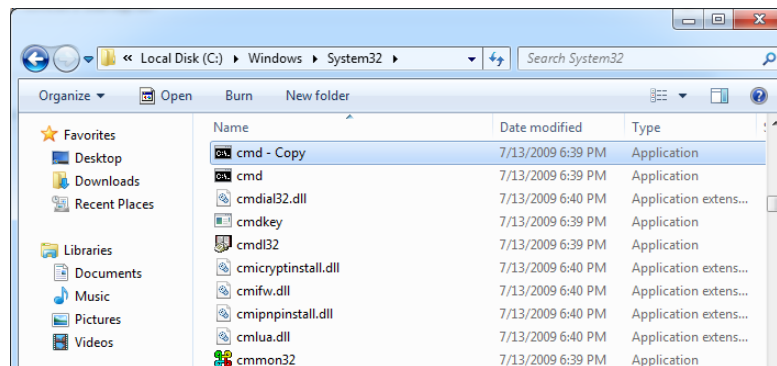
`sudo /etc/init.d/tomcat status`

10.4 How to Create a Windows Service for Jetspeed

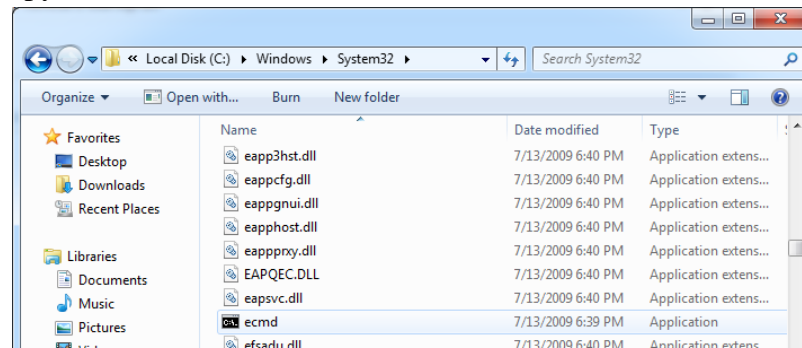
1. To create a windows service for Jetspeed, you need to start/stop the windows service as a system designated *administrator* from the windows command prompt.
2. To do this you can active the “Run as Administrator” option from the command prompt or you can create a separate command prompt that is restricted to administrators. To create a separate command prompt for administrators follow steps 3-10. Otherwise skip top step 11.
3. Go to [my computer](#)



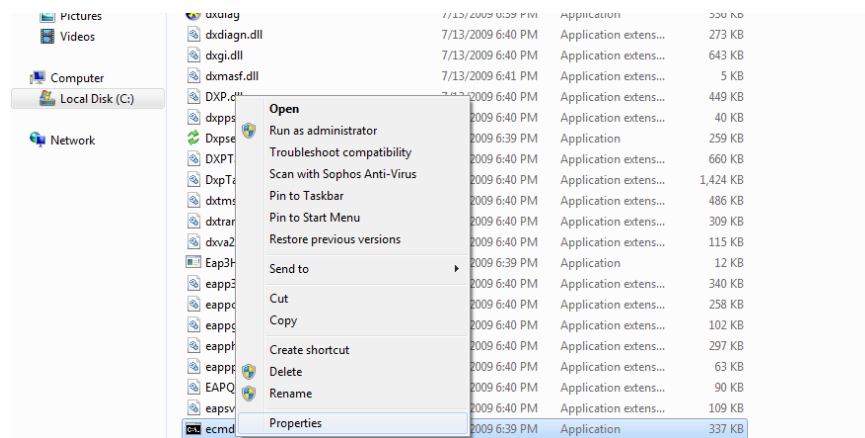
4. From there, go to `C:\Windows\System32` folder
5. Copy `cmd.exe`



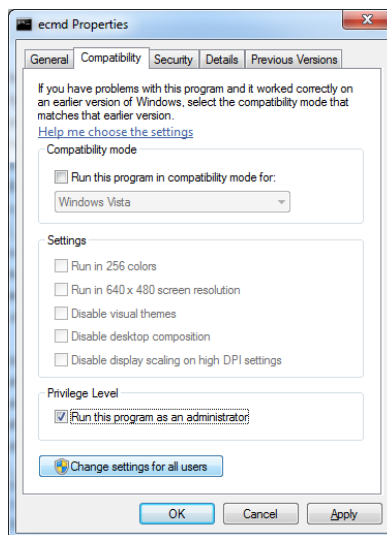
6. Rename the copy to ecmd.exe



7. Right click on ecmd.exe and select properties

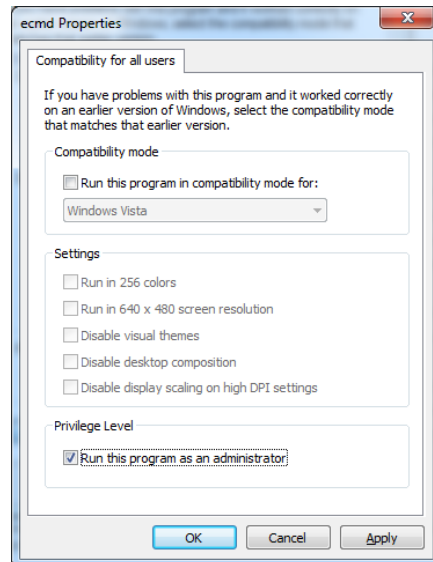


8. Inside properties, check the box, “Run this program as an administrator.”



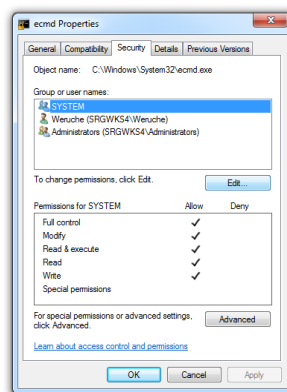
9. Then, go to “Change settings for all users.”

10. In this section, check the box for “Run this program as an administrator” in the “Privilege Level” section.



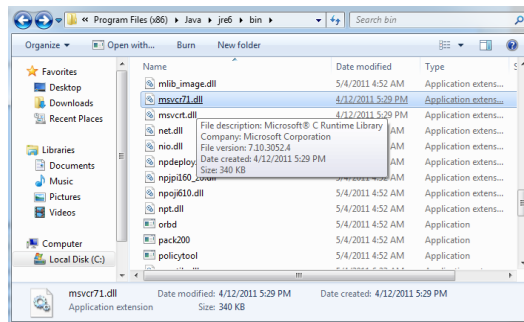
Note: How to Restrict Access of ecmd.exe:

Go to “edit” and “Advanced” to restrict access to ecmd.exe to designated users.

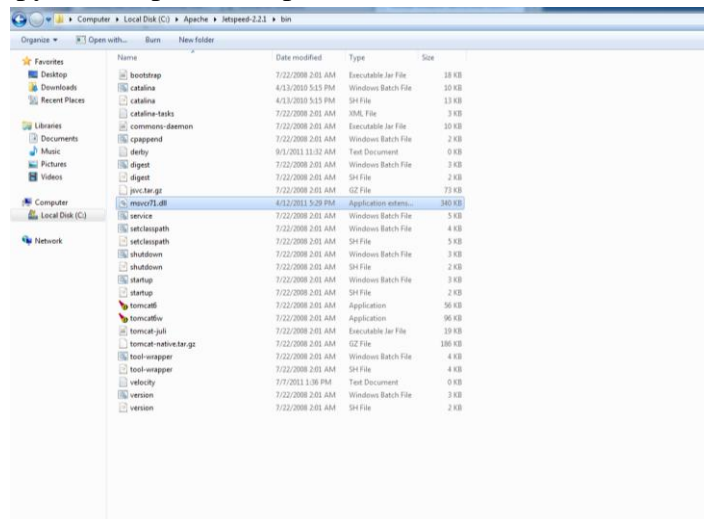


11. Next, go to C:\Program Files (x86)\Java\jre6\bin

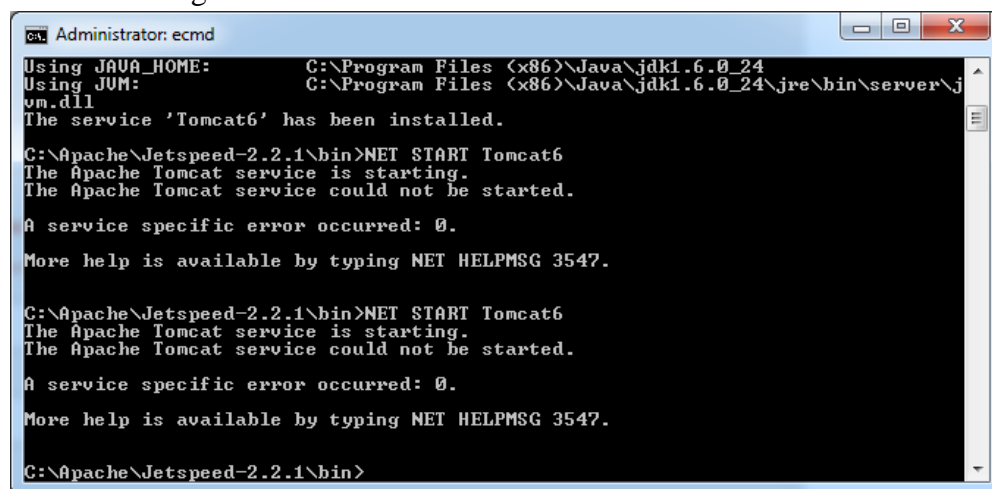
12. There copy msucr71.dll



13. Next, place the copy in C:\Apache\Jetspeed-2.2.1\bin

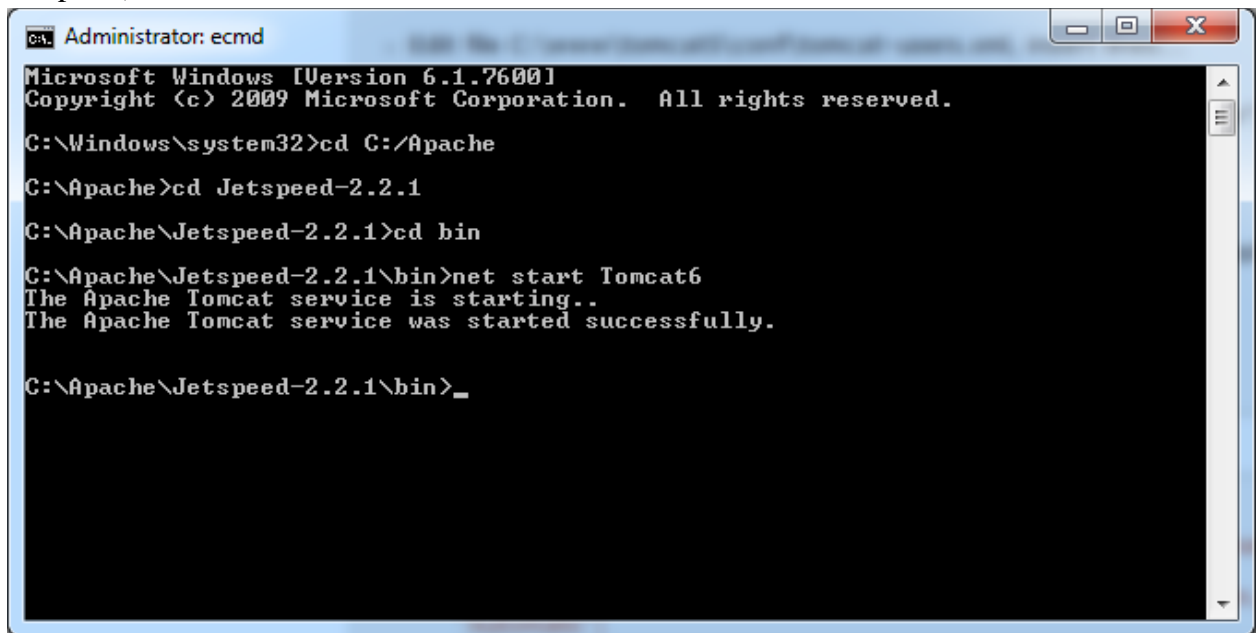


Steps 11-13 are essential to starting the jetspeed service. Without these steps, the service will be installed but you will **not** be able to start it. You will get an error message similar to the one in the image shown below.



14. Next, open the cmd command prompt and run the command *service.bat install* from C:\Apache\Jetspeed-2.2.1\bin. Then, to start the windows service, run the command *NET*

START Tomcat6 (or *NET START Tomcat5*-this command depends on the version of Jetspeed you installed. To check the version, go to C:\Apache\Jetspeed-2.2.1\bin in my computer)



```
Administrator: cmd
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Apache
C:\Apache>cd Jetspeed-2.2.1
C:\Apache\Jetspeed-2.2.1>cd bin
C:\Apache\Jetspeed-2.2.1\bin>net start Tomcat6
The Apache Tomcat service is starting..
The Apache Tomcat service was started successfully.

C:\Apache\Jetspeed-2.2.1\bin>_
```

Command Guide for Command Prompt:

(note: these commands must be executed from C:\Apache\Jetspeed-2.2.1\bin)

Install service	service.bat install
Start service	NET START Tomcat5
Stop service	NET STOP Tomcat5
Uninstall service	service.bat remove

References:

<https://issues.jboss.org/browse/JBPAPP-6126>

<http://www.devside.net/guides/windows/tomcat>

<http://winhelp2002.mvps.org/services.htm>

15. To see a list of the running services, open cmd and enter “tasklist.” Then, press enter.

16. As seen in the image below, **Tomcat6** is now one of the running services

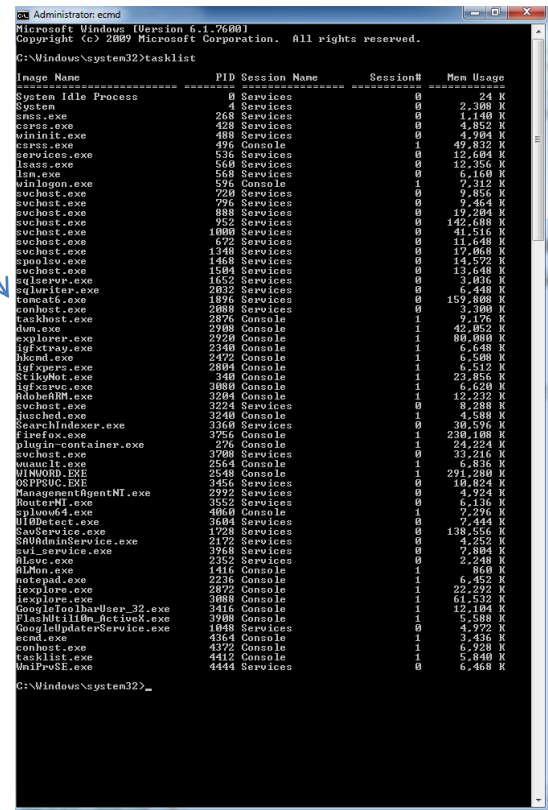
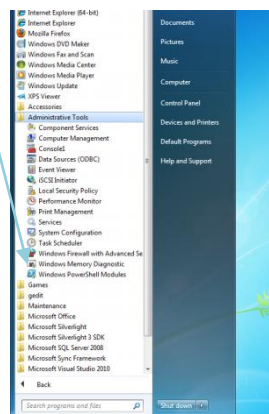
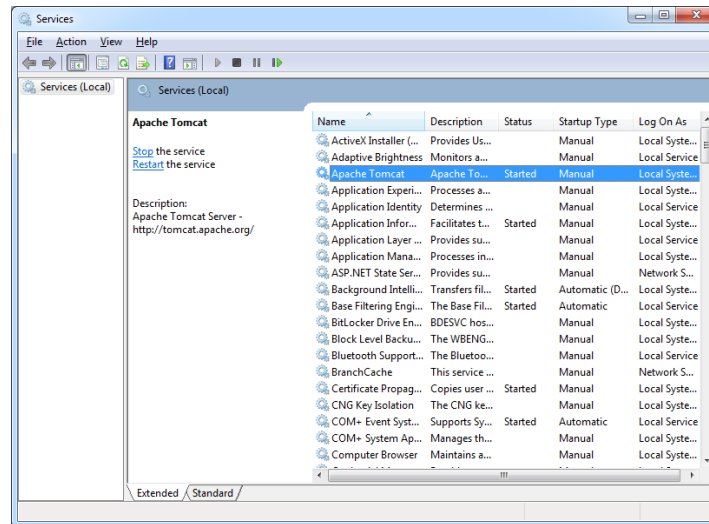


Image Name	PID	Session Name	Session#	Mem Usage
System Idle Process	0	System	0	24 K
System	4	Services	0	2,308 K
smss.exe	268	Services	0	4,149 K
csrss.exe	428	Services	0	4,852 K
wininit.exe	488	Services	0	4,904 K
csrss.exe	496	Console	1	49,832 K
services.exe	536	Services	0	12,604 K
lsass.exe	568	Services	0	12,356 K
lsass.exe	568	Services	0	6,168 K
winlogon.exe	596	Console	1	7,312 K
svchost.exe	728	Services	0	9,856 K
svchost.exe	796	Services	0	9,464 K
svchost.exe	888	Services	0	19,204 K
svchost.exe	952	Services	0	142,684 K
svchost.exe	1008	Services	0	41,516 K
svchost.exe	1172	Services	0	11,648 K
svchost.exe	1348	Services	0	17,068 K
spoolsv.exe	1468	Services	0	14,572 K
svchost.exe	1504	Services	0	13,548 K
sqlservr.exe	1652	Services	0	3,836 K
sqlservr.exe	2832	Services	0	6,448 K
tomcat6.exe	1896	Services	0	159,808 K
conhost.exe	2088	Services	0	3,308 K
taskhost.exe	2376	Console	1	9,172 K
dwm.exe	2308	Console	1	42,852 K
explorer.exe	2328	Console	1	80,888 K
igmpstray.exe	2348	Console	1	6,648 K
hkcmd.exe	2472	Console	1	6,508 K
igmp.exe	2804	Console	1	6,512 K
StikyNot.exe	348	Console	1	23,856 K
igmp.exe	3088	Console	1	6,628 K
AdobeHM.exe	3204	Console	1	12,532 K
svchost.exe	3224	Services	0	8,288 K
tsch.exe	3248	Console	1	4,588 K
SearchIndexer.exe	3368	Services	0	30,596 K
Firefox.exe	3756	Console	1	230,168 K
plugin-container.exe	376	Console	1	24,224 K
svchost.exe	3708	Services	0	33,216 K
ntautcl.exe	2554	Console	1	6,536 K
MINIMORD.EXE	2548	Console	1	291,288 K
OSPFsvc.exe	3456	Services	0	10,424 K
ManagementAgentNT.exe	2972	Services	0	4,924 K
RouterNT.exe	3552	Services	0	6,136 K
plink.exe	4068	Console	1	7,296 K
NIHDetect.exe	3604	Services	0	7,444 K
saoservice.exe	1728	Services	0	138,556 K
SNMPAdminService.exe	2172	Services	0	4,252 K
swi_service.exe	3968	Services	0	7,804 K
ALSec.exe	2352	Services	0	2,448 K
ALMon.exe	1416	Console	1	868 K
notepad.exe	2236	Console	1	6,452 K
explore.exe	2872	Console	1	22,292 K
explore.exe	3088	Console	1	61,532 K
GoogleIshUser_32.exe	3416	Console	1	12,184 K
FlashMutilion_ActiveX.exe	3908	Console	1	5,588 K
GoogleUpdaterService.exe	1048	Services	0	4,772 K
cmd.exe	4354	Console	1	3,436 K
conhost.exe	4372	Console	1	6,928 K
tasklist.exe	4412	Console	1	5,848 K
UnifProSE.exe	4444	Services	0	6,468 K

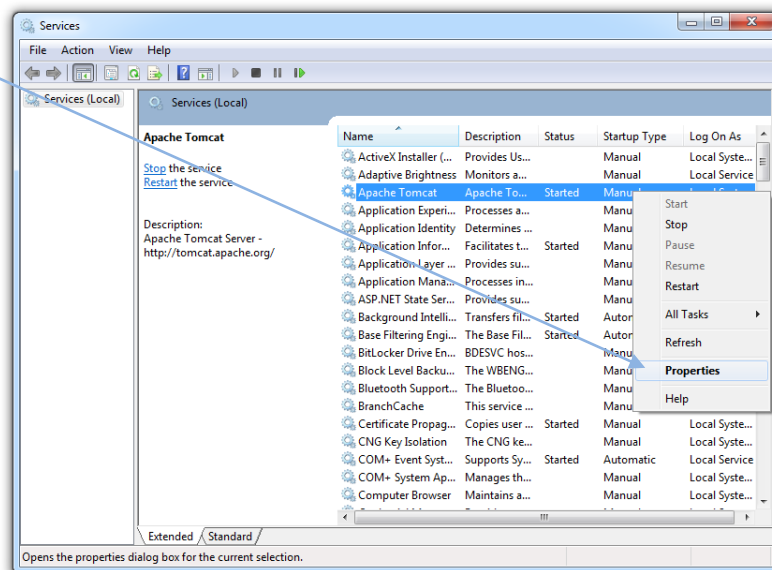
17. To have the service start at system startup, go to the start menu. Then, go to “All Programs.” From the list of programs, find “Administrative Tools”. From this list, you will see the program, **Services**, as shown in the image below.



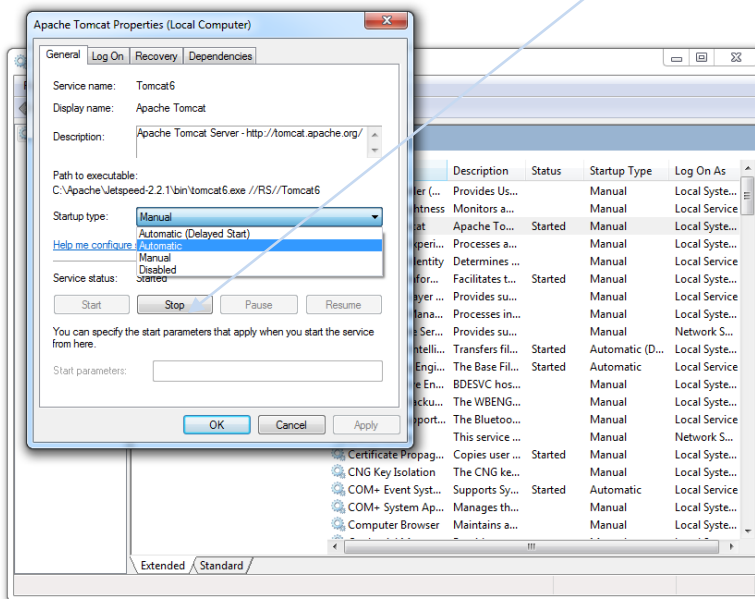
18. Open Services. As shown in the image below, the status of the Tomcat (Jetspeed) service is listed as started but this service is a manual process instead of an automatic process.



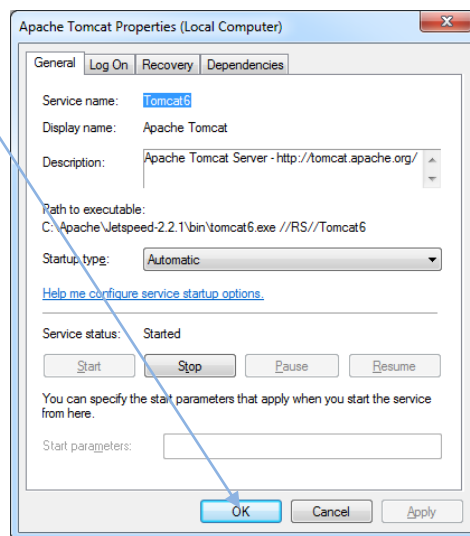
19. To change this to an automatic service. Right click on “Apache Tomcat” and select Properties



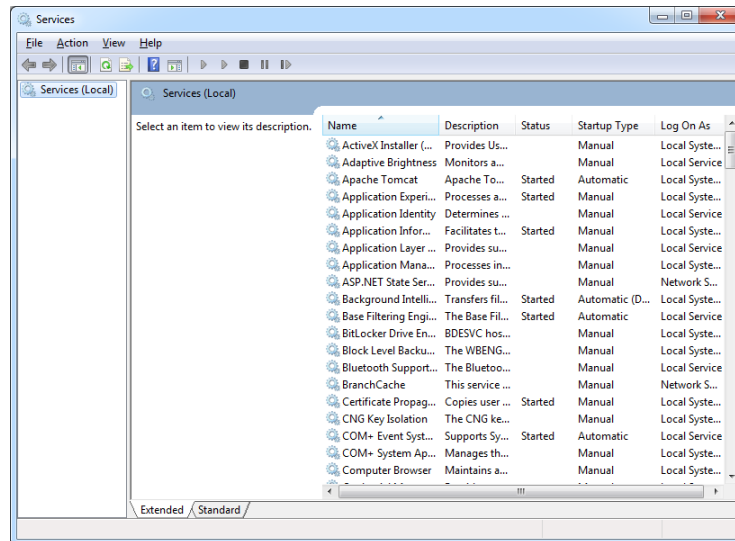
20. This will open a new window. To change the Startup type, open the drop down menu as show in the image below. Select the **Automatic** option.



21. Then, press **OK**.



22. As shown in the image below, the Tomcat (Jetspeed) service is now an automatic startup service.



11 Jetspeed Setup

11.1 How to Create a Portal Space

1. Under the tab for Public Space is the option “Add Space.” Click on it to create a new portal space

The screenshot shows the Jetspeed Public Space configuration page. The left sidebar has a 'Public Space' tab selected, and the 'Add Space' option is highlighted. The main content area displays the 'User Registration' form with fields for Username, Email Address, Password, and various optional fields like First Name, Last Name, Department, etc. Below the form is the 'Role Security Test' table.

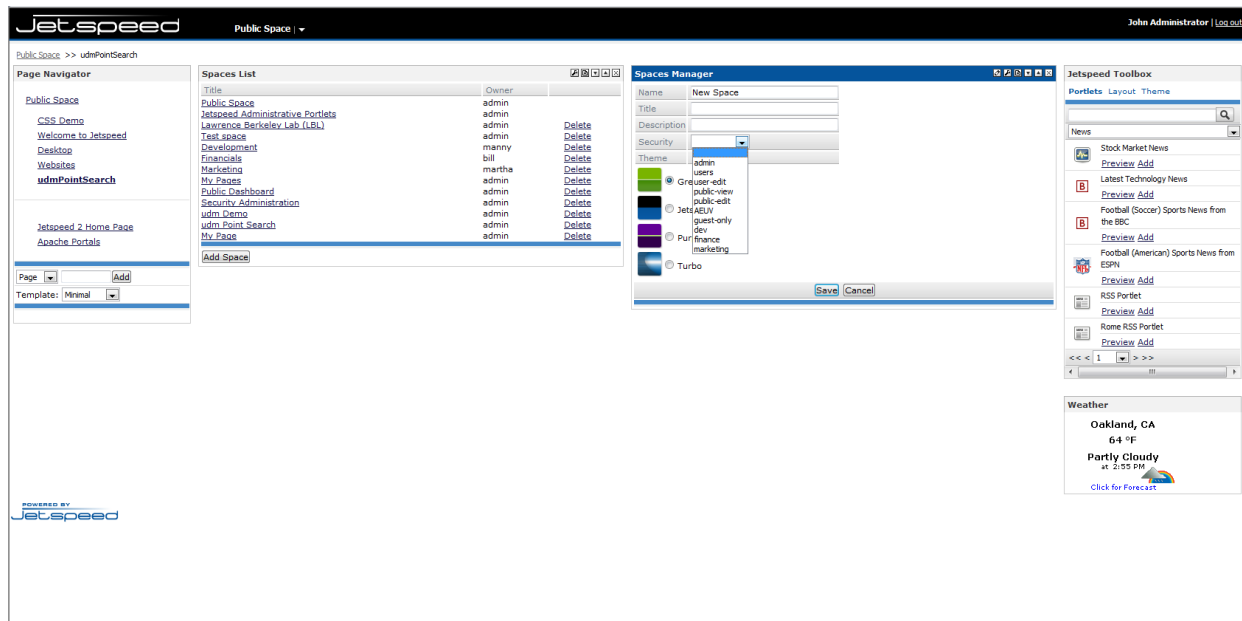
Role	Ref	Name	Role	PortletRequest	ServletRequest
Administrator	admin	true	true		
Manager	manager	false	false		
User	user	true	true		

2. Once there, you have various options for editing and creating portals

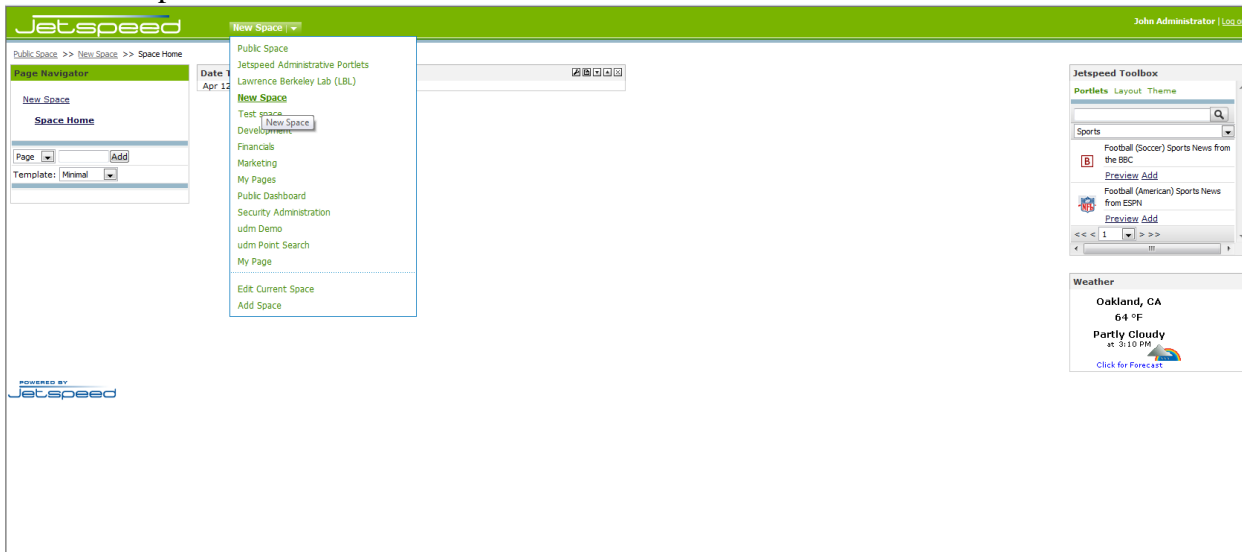
The screenshot shows the Jetspeed Spaces Manager page. The left sidebar has a 'Public Space' tab selected, and the 'Add Space' option is highlighted. The main content area displays the 'Spaces Manager' form with fields for Name, Title, Description, Security, and Theme. Below the form is the 'Spaces List' table.

Title	Owner	Delete
Public Space	admin	
Jetspeed Administrative Portlets	admin	Delete
Lawrence Berkeley Lab (LBL)	admin	Delete
Test space	admin	Delete
Development	manny	Delete
Financials	bill	Delete
Marketing	martha	Delete
My Pages	admin	Delete
Public Dashboard	admin	Delete
Security Administration	admin	Delete
udm Demo	admin	Delete
udm Point Search	admin	Delete
My Page	admin	Delete

- Click on “Add Space” to create a new portal. From there you can name and select the level of security for the portal. In the following example, the new portal has been called “New Space.”

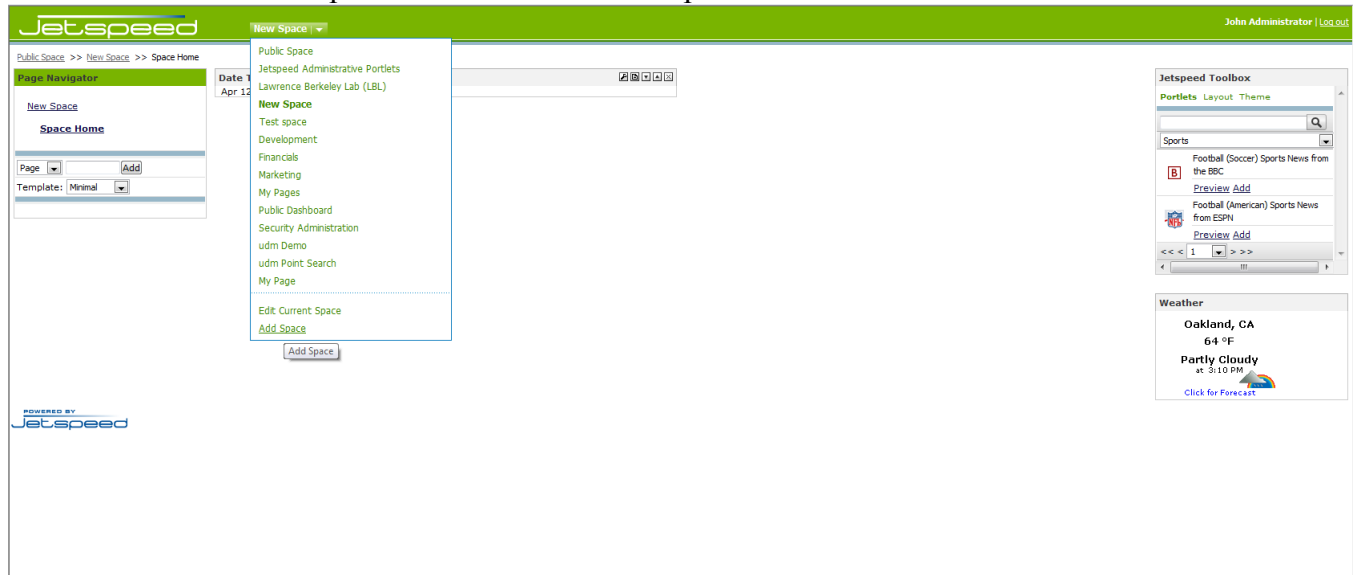


- Click “Save.” The new portal has now been created. From here you can modify your new portal as desired.

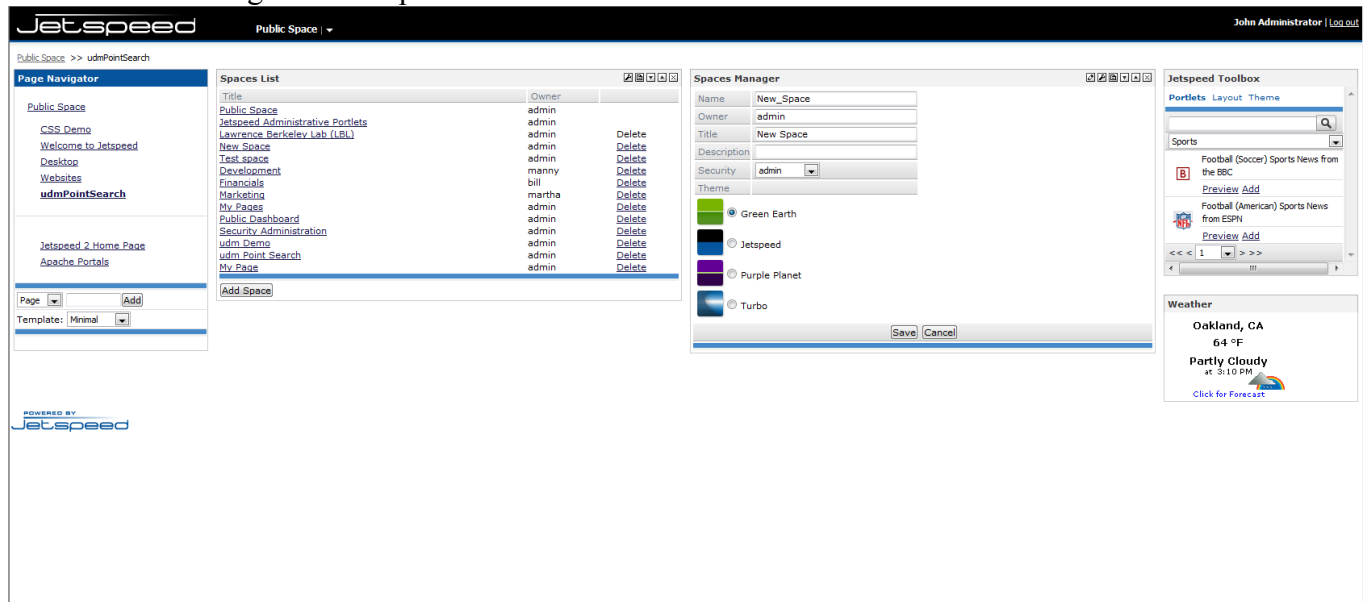


11.2 How to Delete a Portal Space

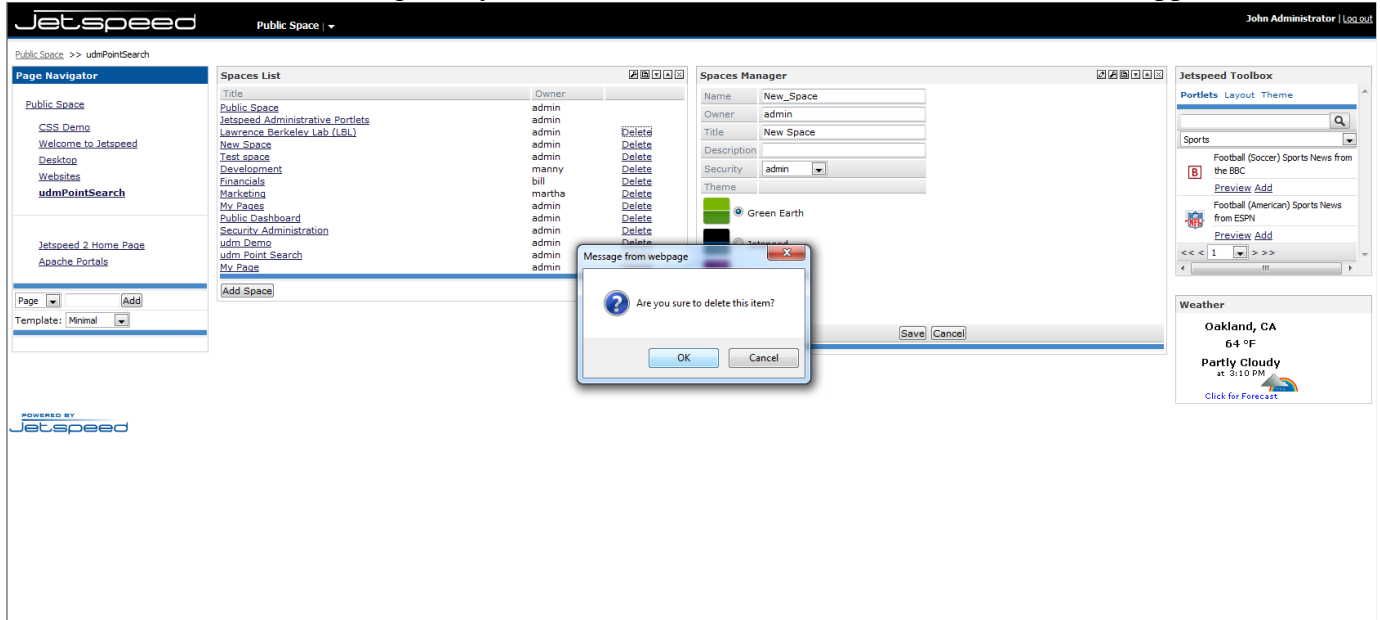
1. Go to the Tab for portals and click on “Add Space”



2. Once there go to the “Spaces List”



3. Select “Delete” for the portal you want to delete and a confirmation window will appear.



4. Once deleted, the portal will no longer appear on the “Spaces List.”

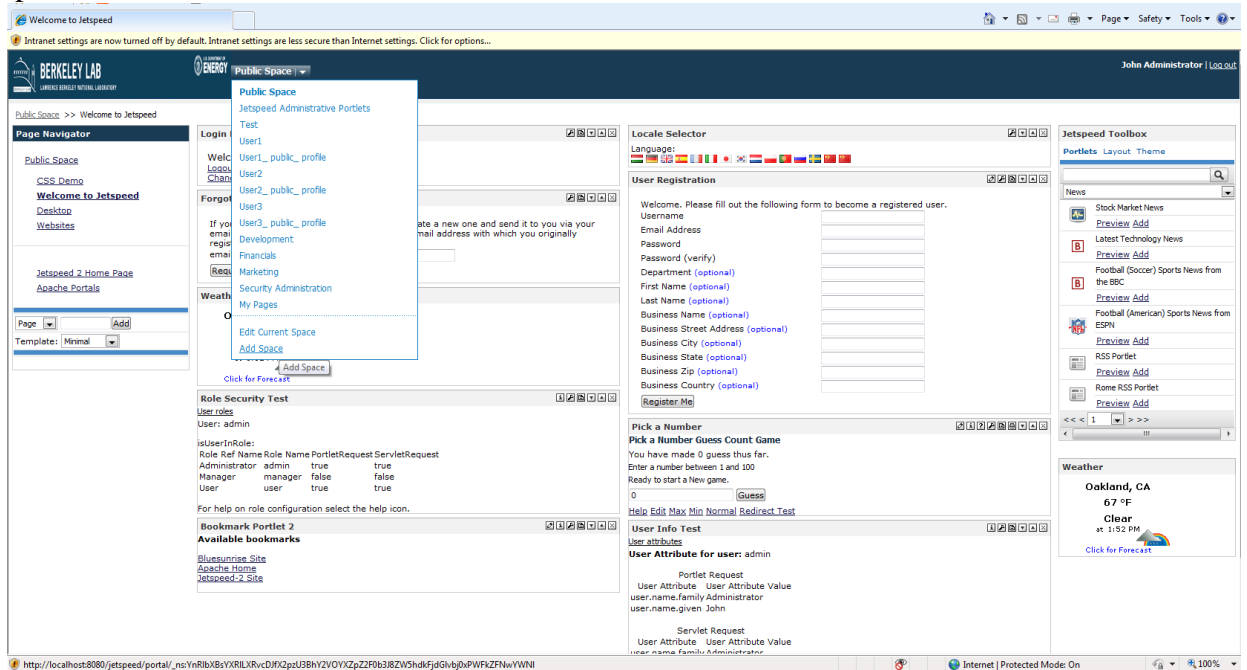
11.3 How to Generate Various Security Settings in Jetspeed

Throughout this guide User1, User2, and User3 will refer to the new users created. The following pages refer to the new pages created:

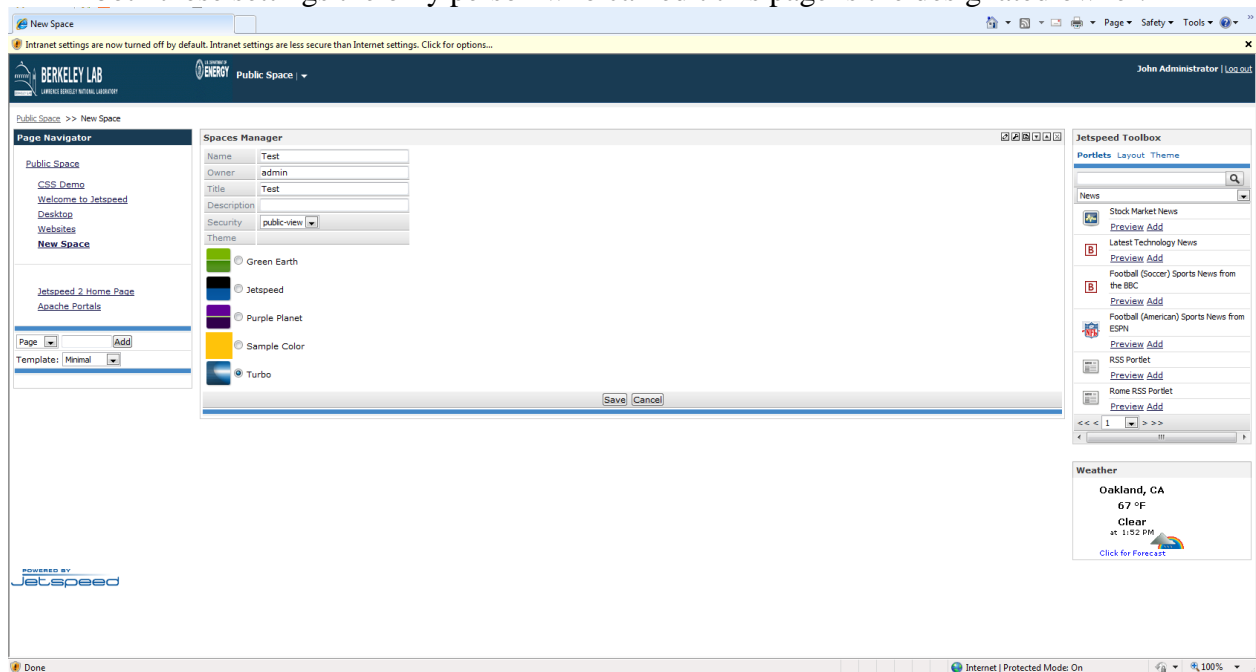
Public access, restricted edit	Restricted access, Restricted edit
Test: visible to everyone but only User1 can edit this page	User1: User1’s private profile-only User1 can view and edit this page
User1_public_profile: User1’s public profile-visible to everyone but only User1 can edit this page	User2: User2’s private profile- only User2 can view and edit this page
User2_public_profile: User2’s public profile-visible to everyone but only User2 can edit this page	User3: User3’s private profile-only User3 can view and edit this page
User3_public_profile: User3’s public profile-visible to everyone but only User3 can edit this page	

11.4 How to create a page that only one user can edit, but everyone else can view

1. Under the tab for Public Space is the option “Add Space.” Click on it to create a new space

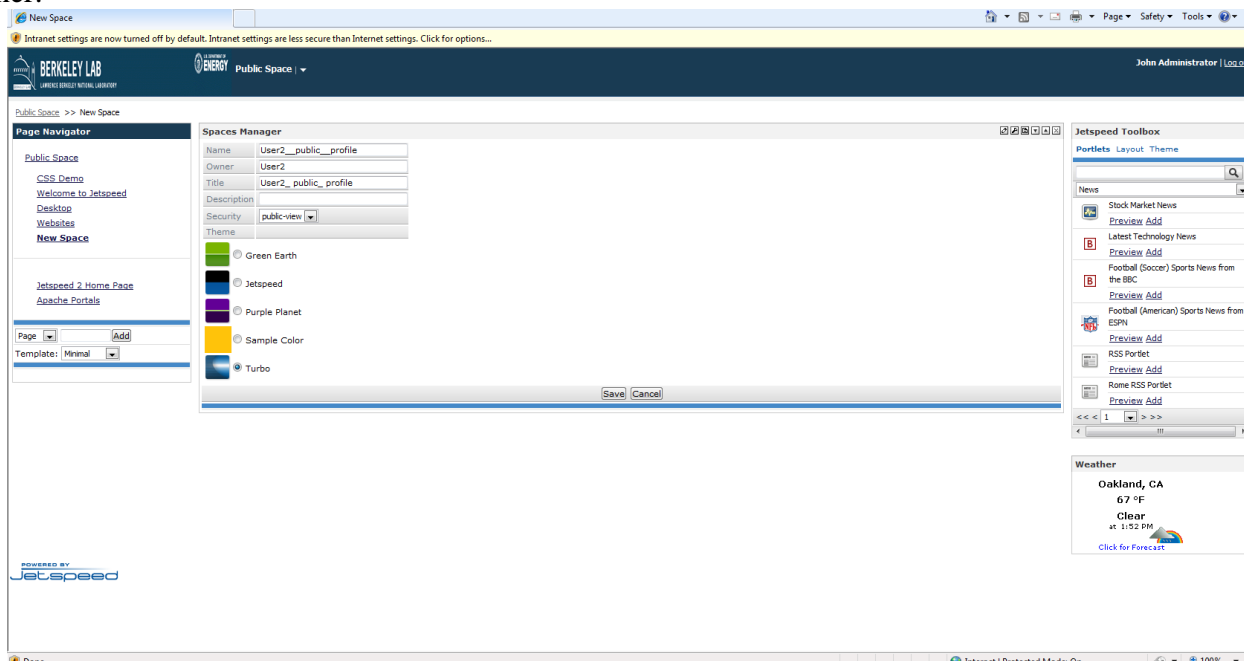


2. After this page has been created, go to “Edit Current Space.” Set the owner to the user that will be the one to edit this page. In the screenshot below, this user is User1. Then, set the security of the page—“public-view” allows everyone to see this page without logging into jetspeed, while, “user-view” allows only registered users to view that page. With both these settings the only person who can edit this page is the designated owner.



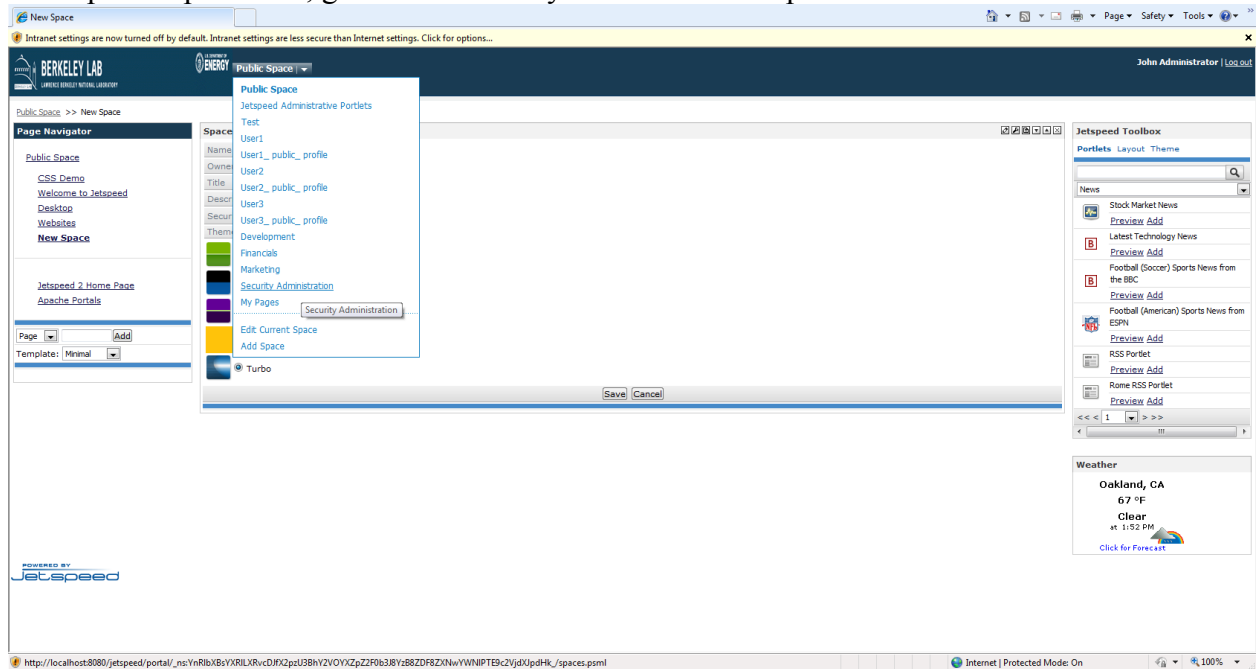
11.5 How to create a page for a user that only he can edit, but everyone else can view

Repeat step 1. After a page has been created for this user, go to “Edit Current Space.” Set the owner to the user that will be the one to edit this page. In the screenshot below, this user is User2. Then, set the security of the page—“public-view” allows everyone to see this page without logging into jetspeed, while, “user-view” allows only registered users to view that page. With both these settings the only person who can edit this page is the designated owner.

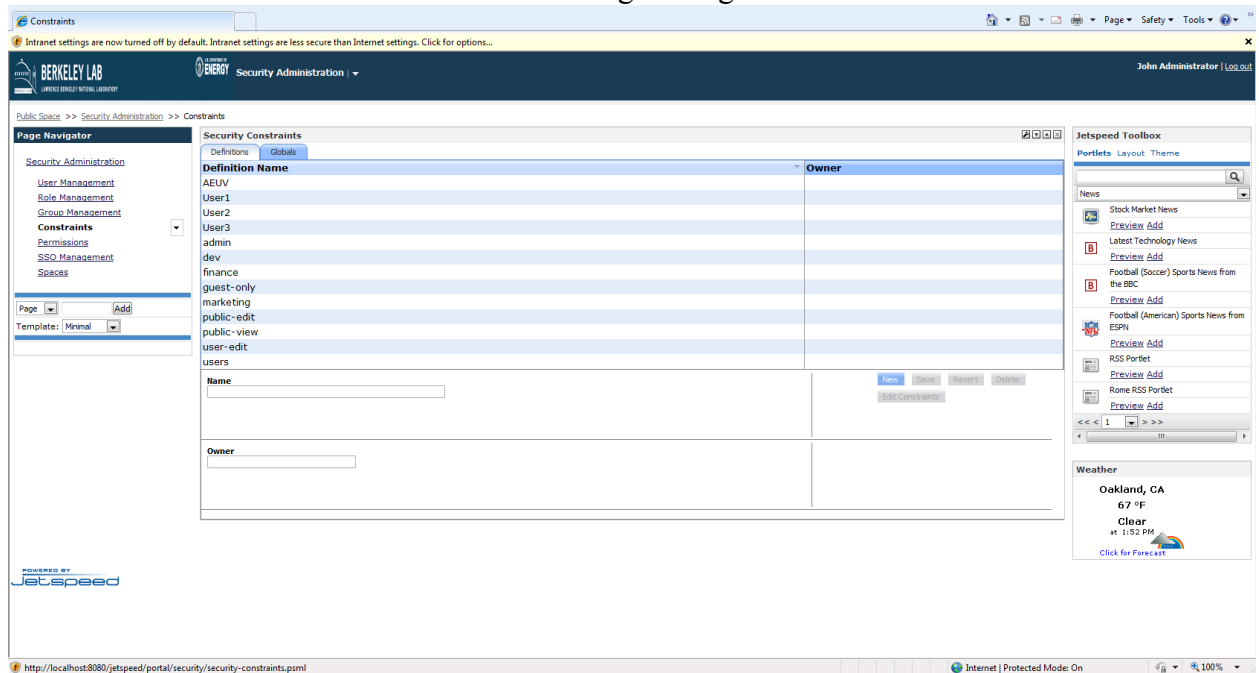


11.6 How to create a page for a user that only he can view and edit

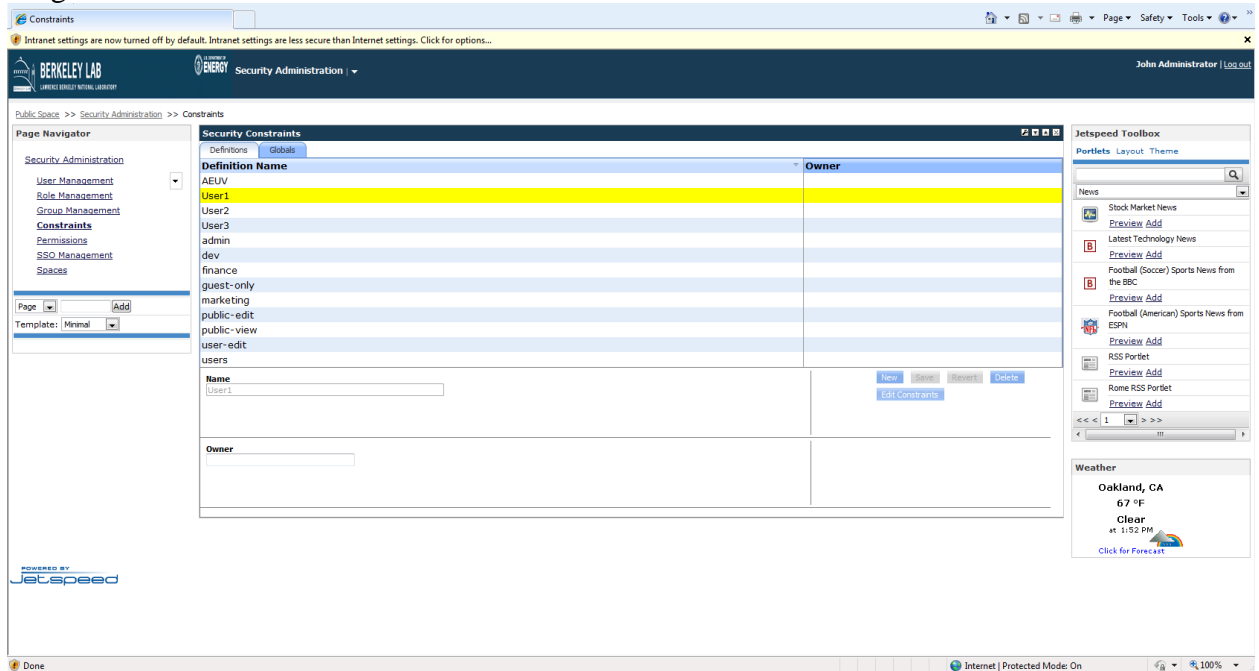
1. Repeat step 1. Then, go to the “Security Administration” portal in the menu.



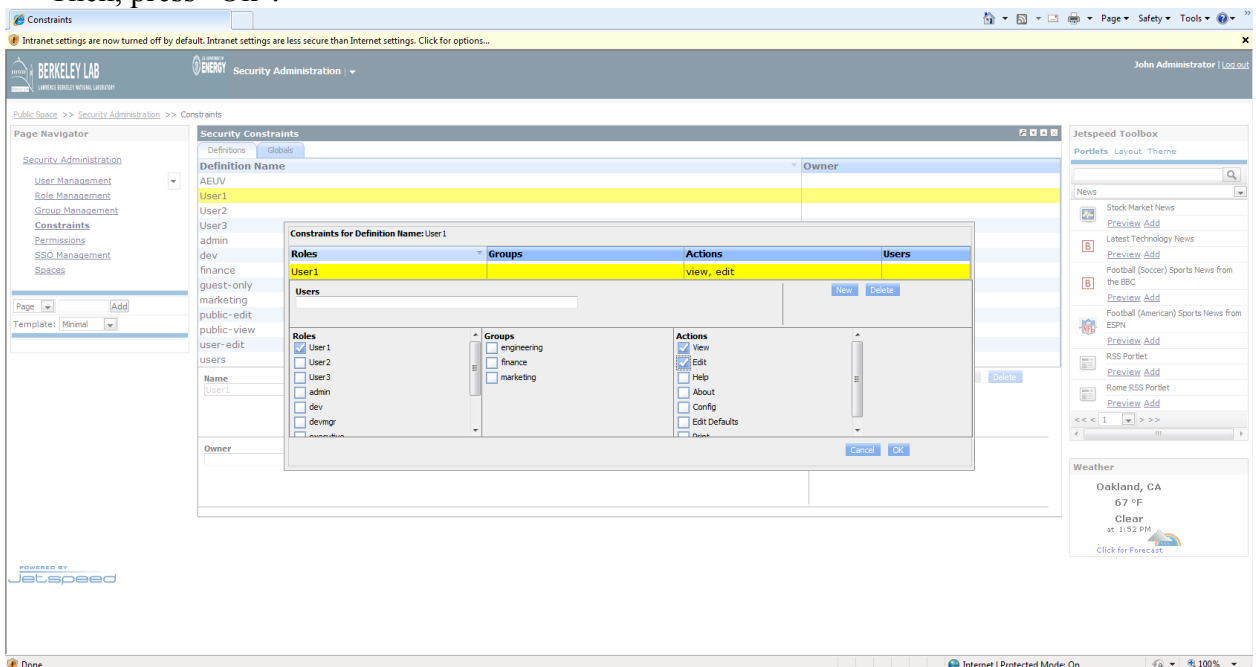
2. Once there click on “Constraints” in the “Page Navigator” menu.



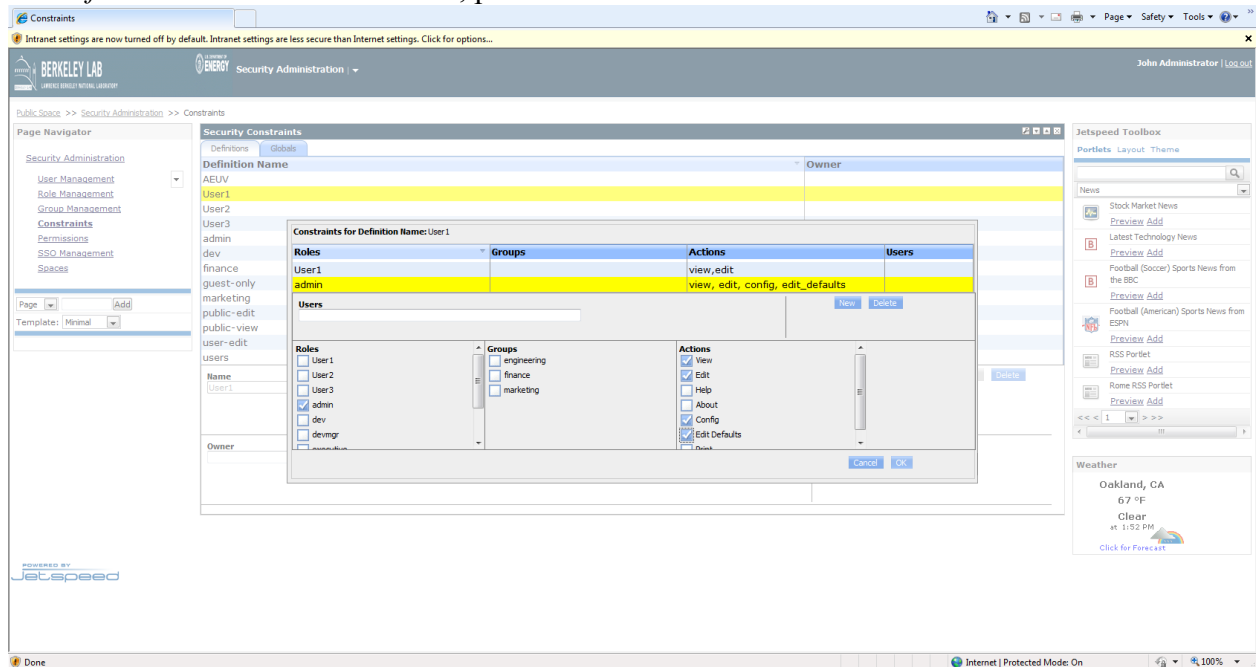
- Then, create a security constraint for this user. In this example, User1 will be used. At the bottom of the page is the section “Name.” Press “New”, and then enter in the name User1 and press “Save.” This name will now be show in the list of security constraints as seen above.
- Then, click on this new security constraint and press “Edit Constraints” as seen in the image below.



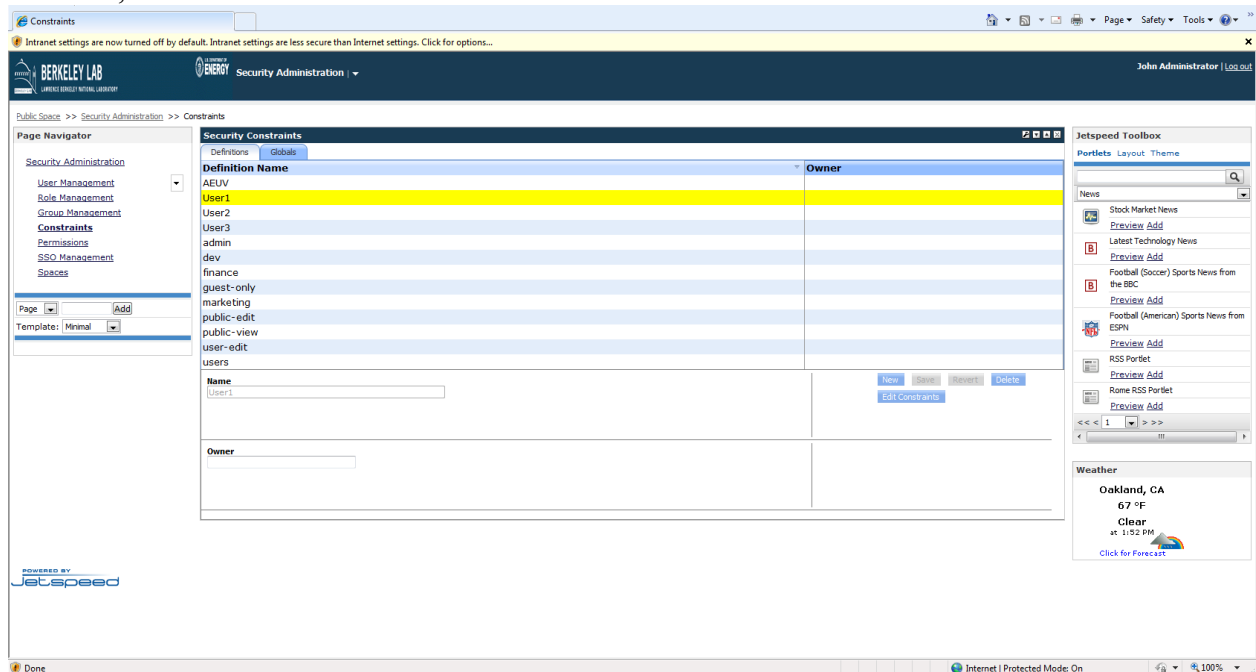
- Then, the menu in the image below should appear.
- First, press “New” and select *User1* under “Roles” and *View* and *Edit* under “Actions.” Then, press “Ok”.



7. Next, press “New” and select *Admin* under “Roles” and *View, Edit, Config, and Edit Defaults* under “Actions.” Then, press “Ok”.

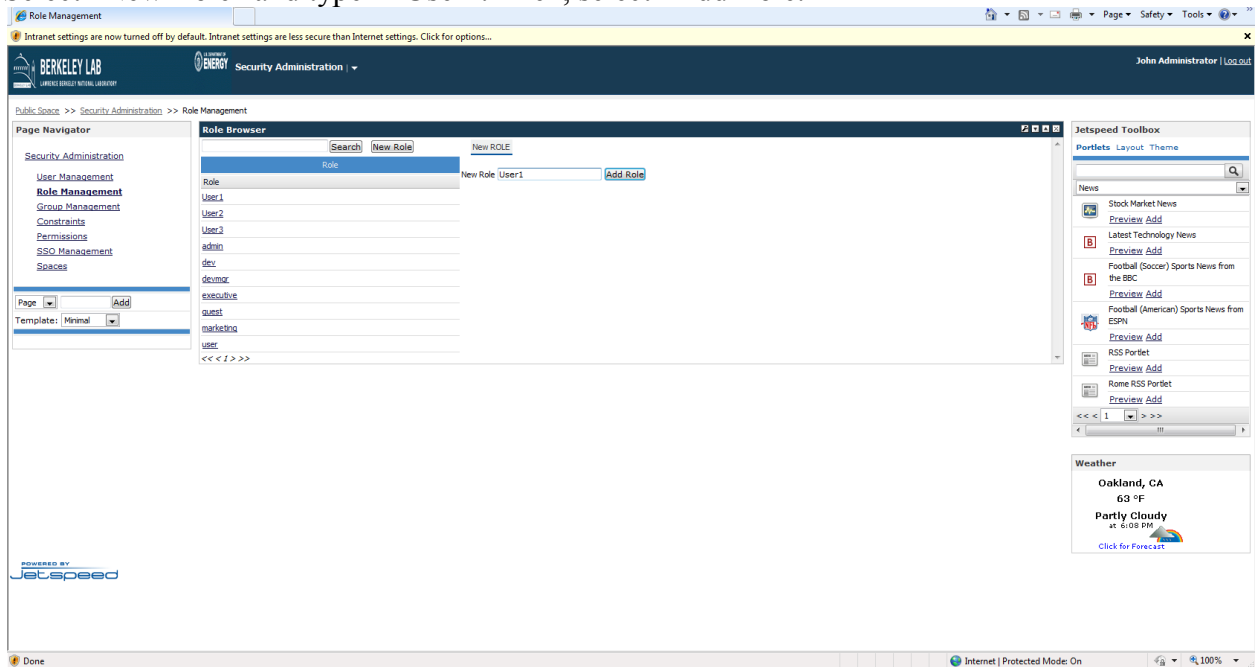


8. Then, click on “Save” in the main menu.

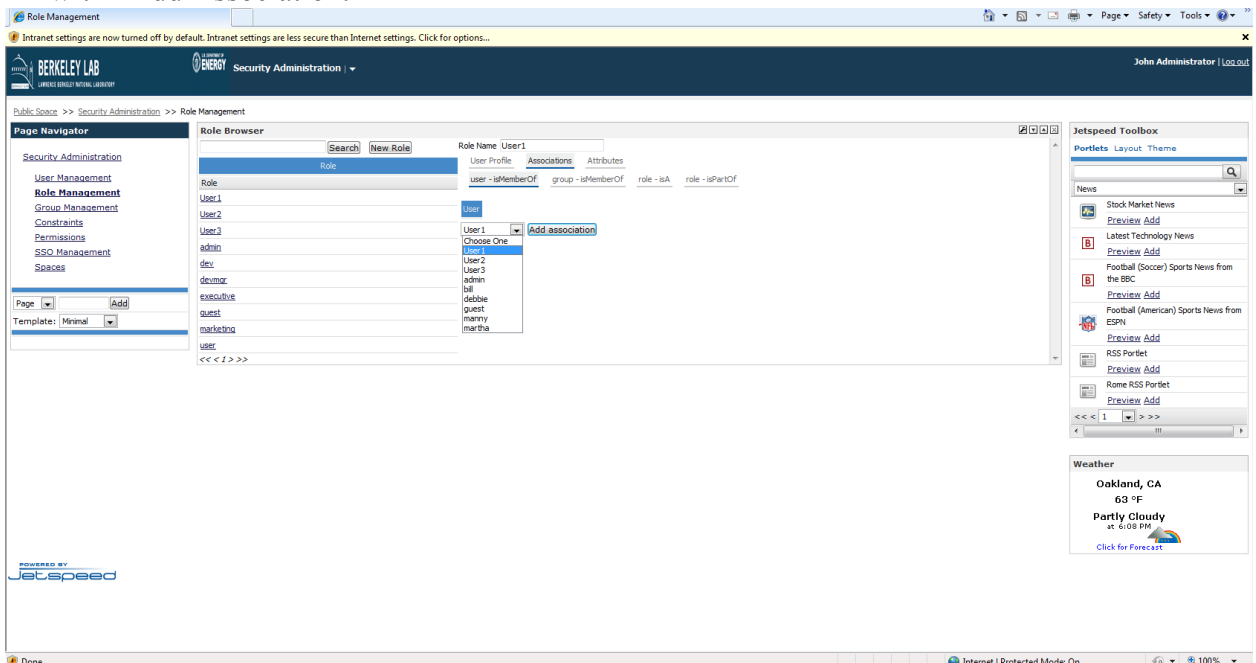


9. Then, click on “Role Management” in the “Page Navigator” menu.

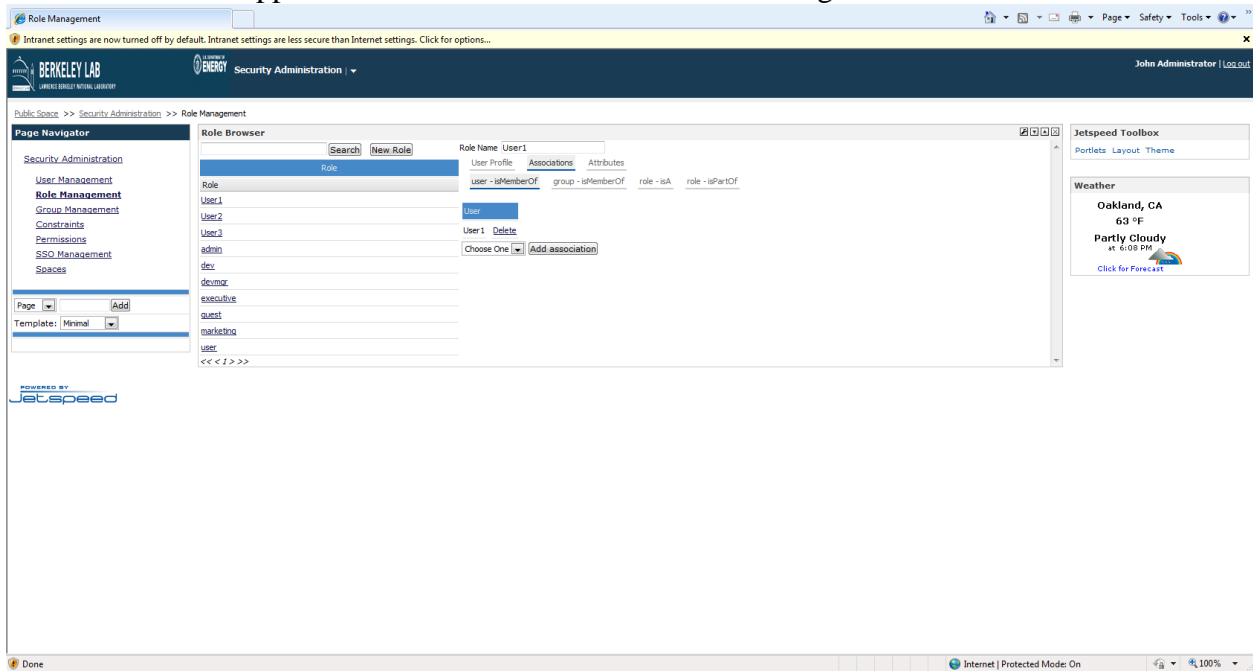
10. Select “New Role” and type in User1. Then, select “Add Role.”



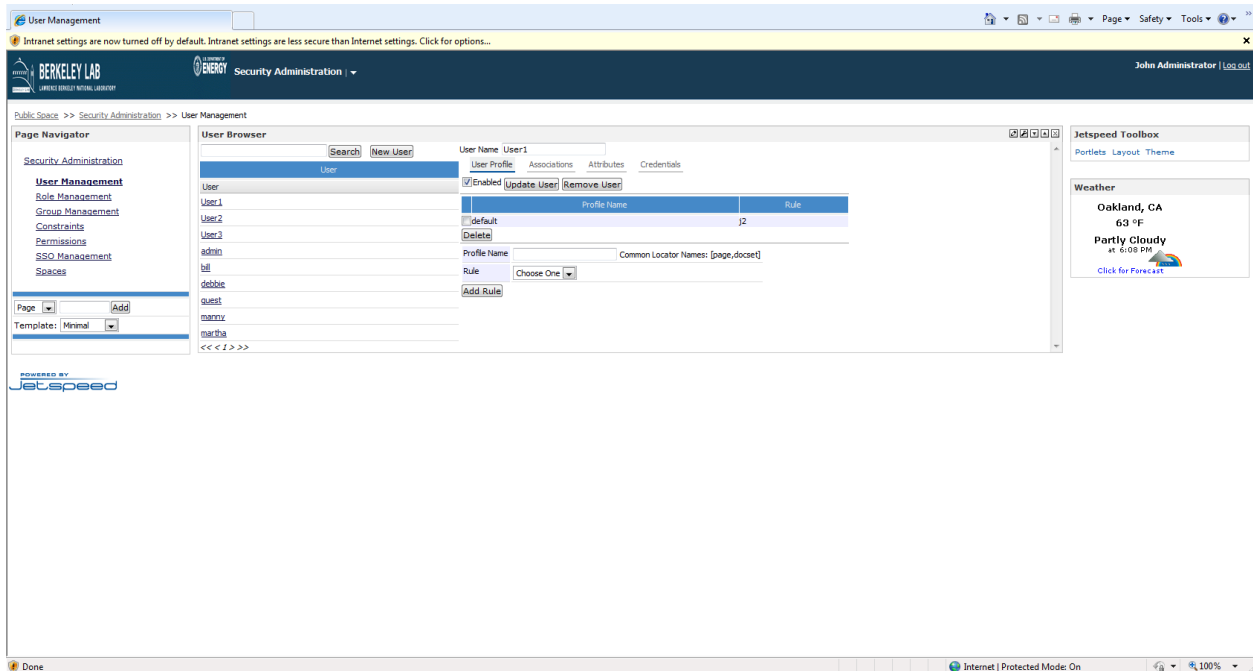
11. After this role has been created and is listed in the list of roles as seen above, select User1 from the list. Once User1 appears in “Role Name,” (shown below) select “Associations.” Then, from the drop down menu shown below, select User1. Then add it to the user list with “Add Association.”



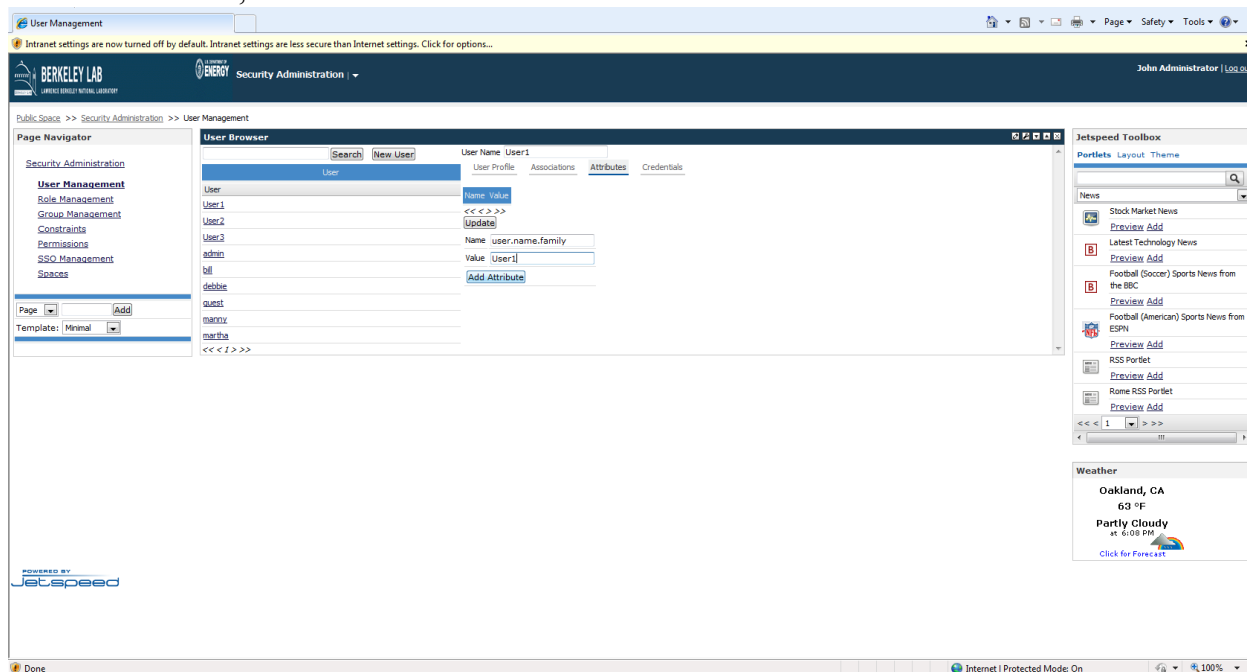
12. User1 now appears on the list of users as seen in the image below.



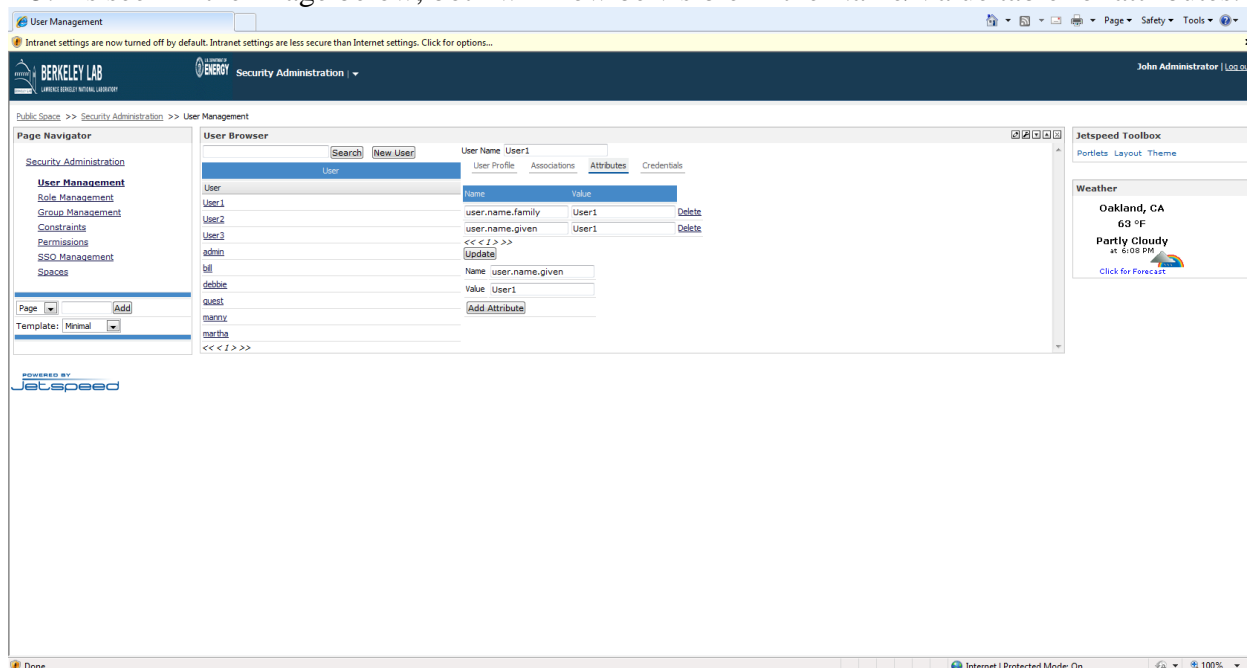
13. Next, go to “User Management” in the “Page Navigator” menu. From the list of users, select User1.



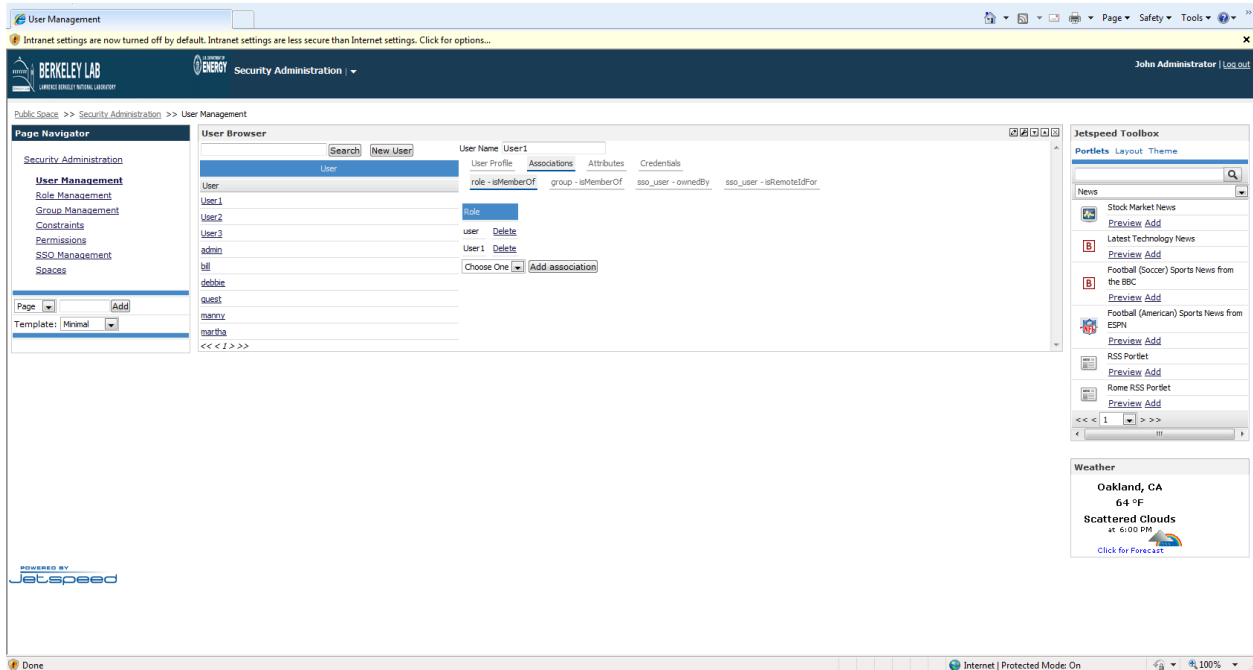
14. Then, select “Attributes” and enter **user.name.family** for “Name” and **User1** for “Value.” Then, select “Add Attribute.” Next, enter **user.name.given** for “Name” and **User1** for “Value.” Then, select “Add Attribute.”



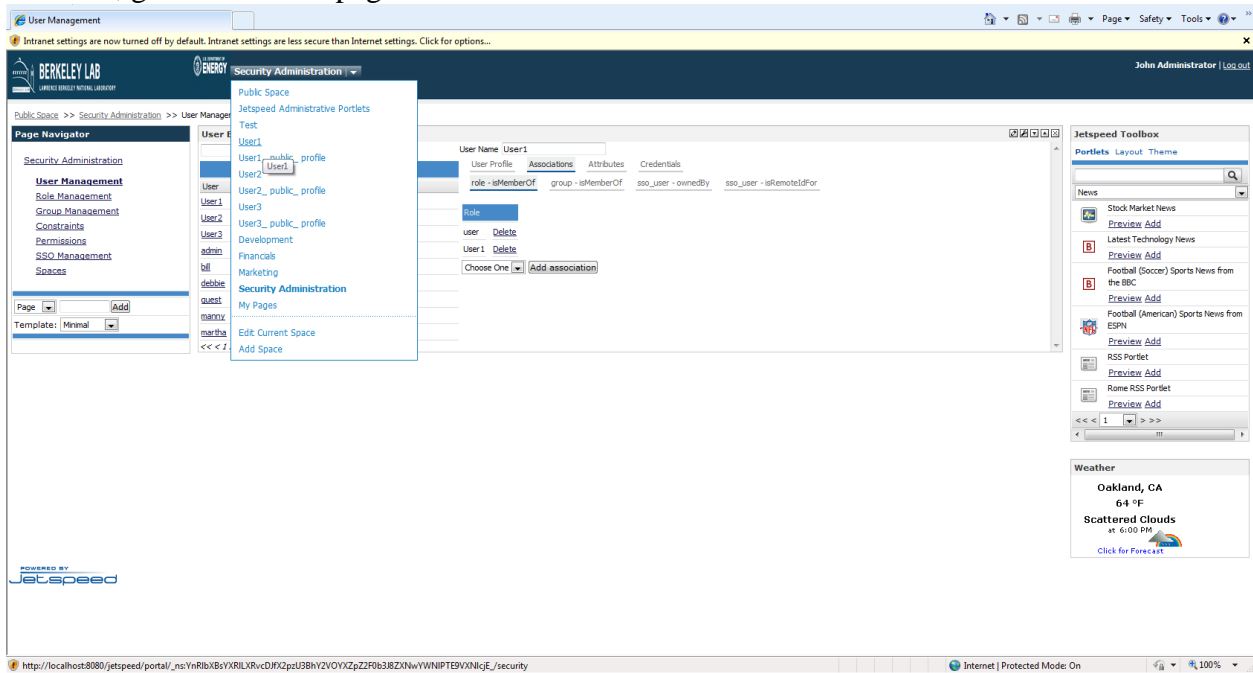
15. As seen in the image below, both will now be visible in the Name/Value table for attributes.



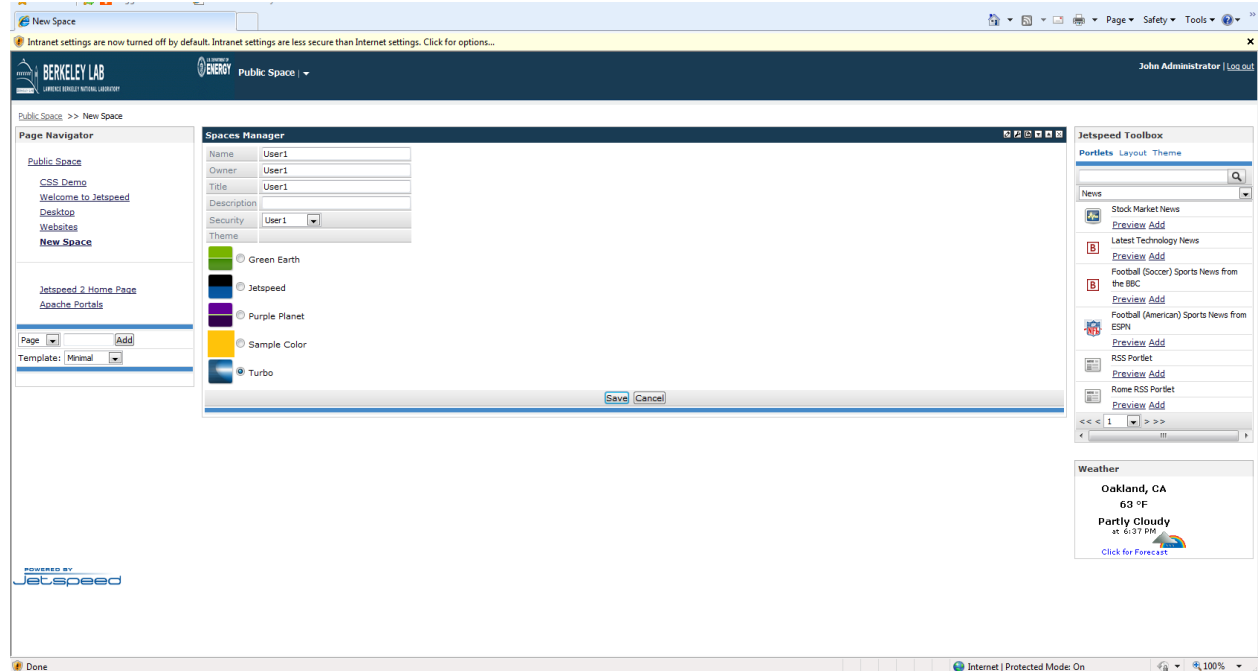
16. Next, go to “Associations” and select User1 from the associations shown in the drop down menu for “Add Associatsion.” Once selected, the role User1 will apperar in the list of roles for User1 as seen below.



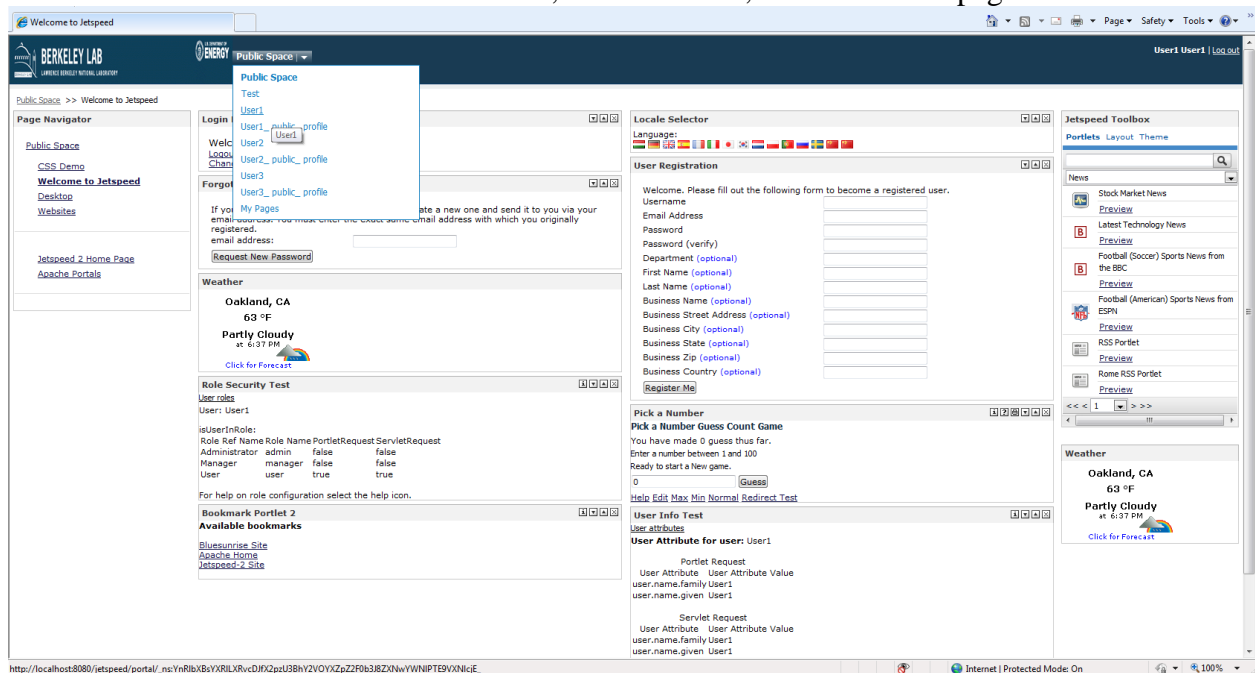
17. Then, go to the User1 page

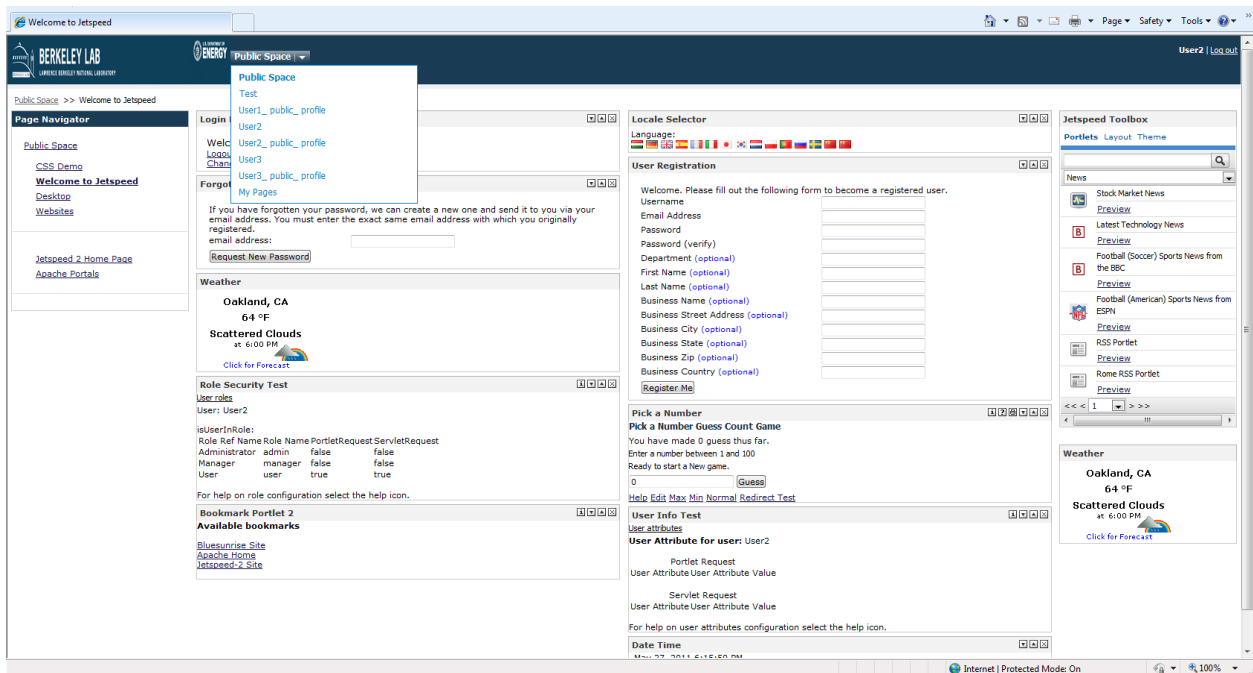
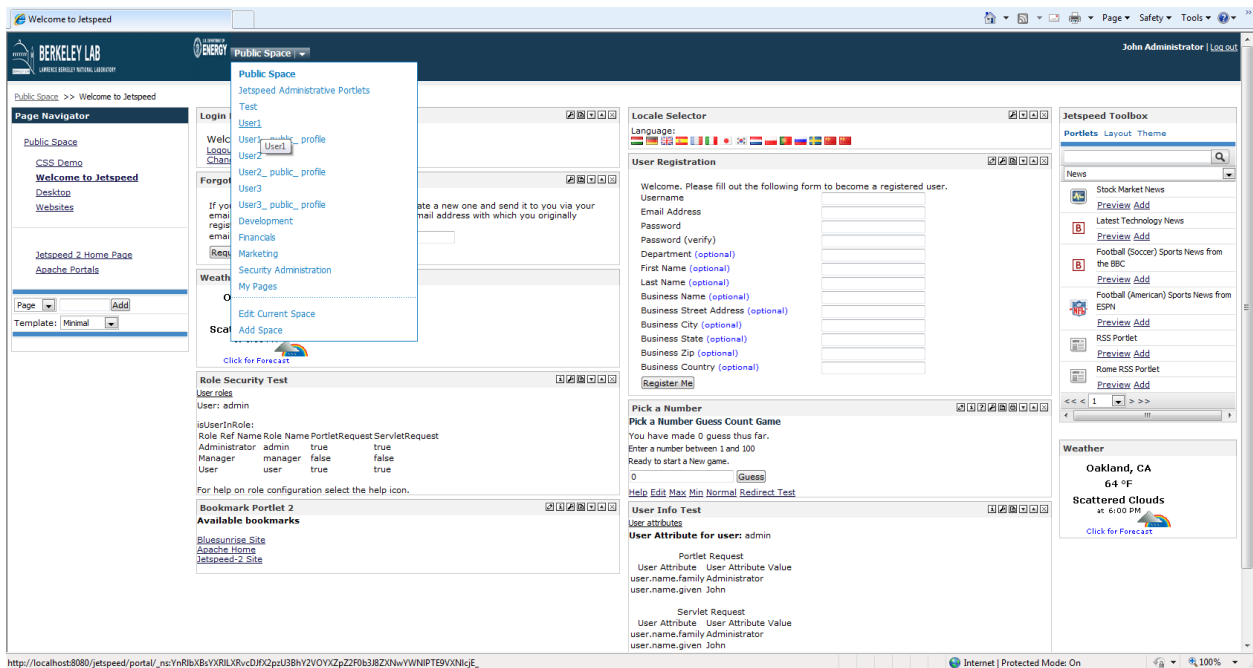


18. Once there, go to “Edit Current Space” and set the security for this page to be User1 as seen below.



19. As show in the image below, User1 and the admin are the only users who can view this page. While the other users, such as User2, cannot see this page.





11.7 How to Insert a Company Logo into Jetspeed

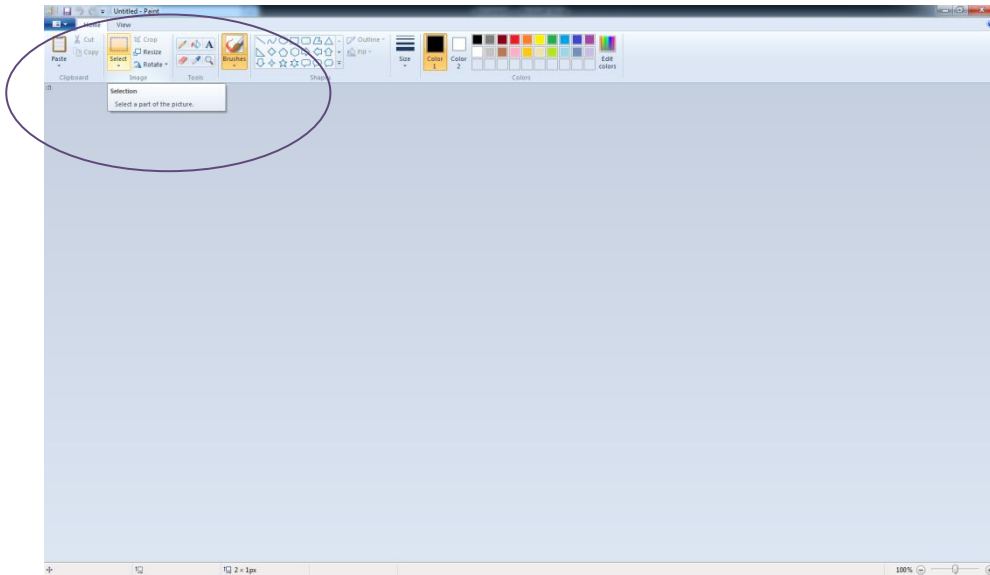
This document will discuss the following items:

- i. How to convert an image or picture to a GIF image or picture
- ii. How to insert a company logo into Jetspeed

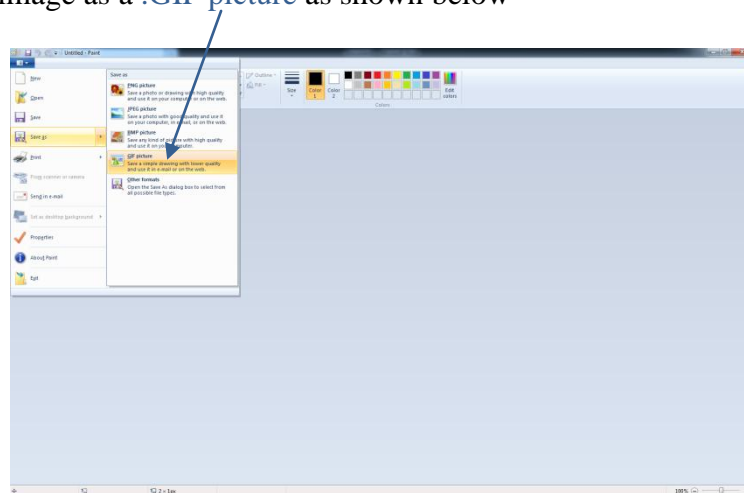
- iii. How to change the header background color for the new header logo

11.7.1 How to convert an image or picture to a .GIF image or picture

1. Convert the company logo into a GIF image using Windows Paint or another image converter.
 - a. If windows paint is used proceed to step 2, otherwise, skip to step 4.
2. Next, crop the logo to the size you want using “Select”



3. Then, save the image as a .GIF picture as shown below

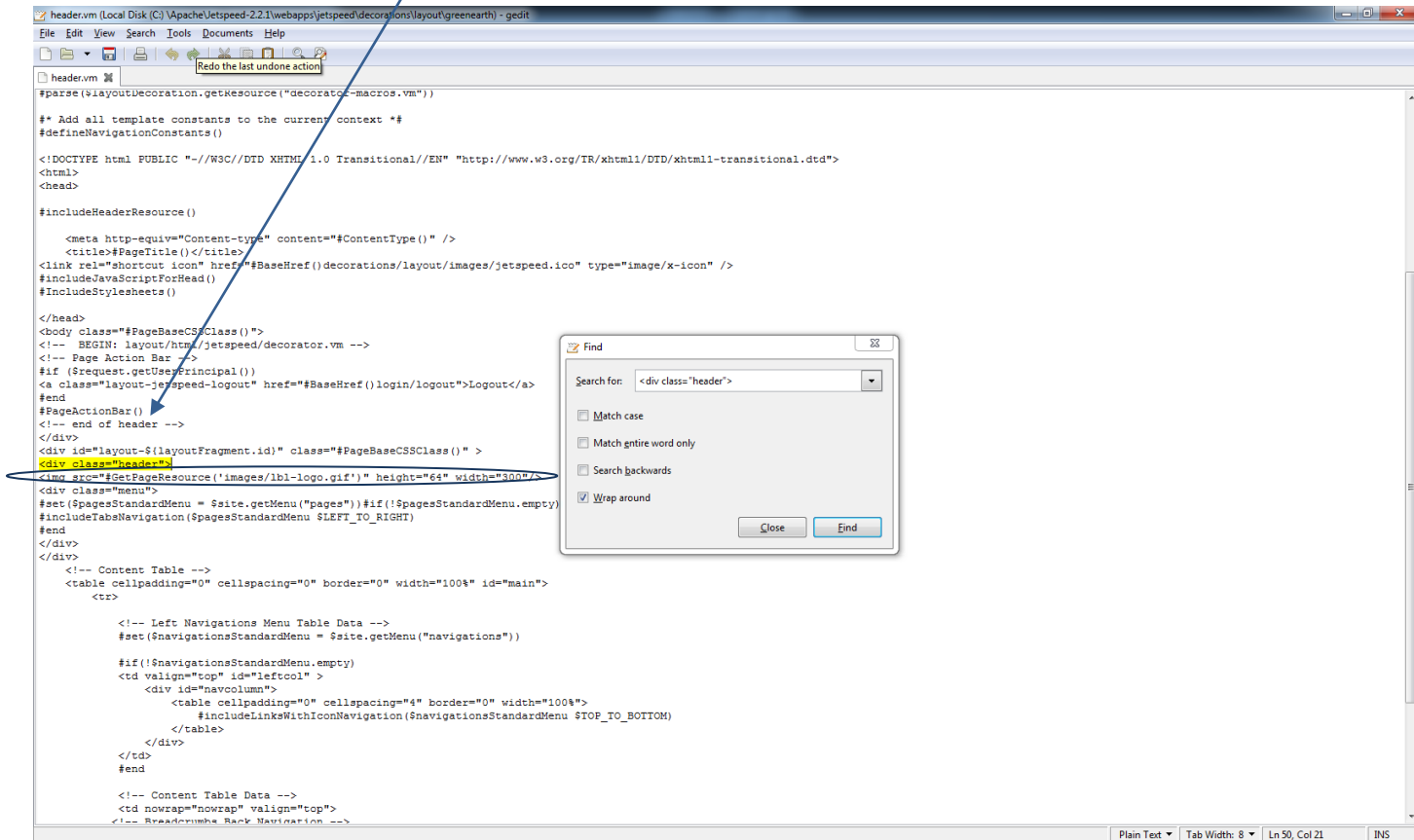


11.7.2 How to insert a company logo into Jetspeed

4. Next, insert the company logo (now a GIF image) into the following directories:
 - a. C:\Apache\Jetspeed-2.2.1\webapps\jetspeed\decorations\layout\purpleplanet\images

- b. C:\Apache\Jetspeed-2.2.1\webapps\jetspeed\decorations\layout\greenearth\images
- c. C:\Apache\Jetspeed-2.2.1\webapps\jetspeed\decorations\layout\jetspeed\images
- d. C:\Apache\Jetspeed-2.2.1\webapps\jetspeed\decorations\layout\turbo\images

5. Then, search for **<div class="header">** inside C:\Apache\Jetspeed-2.2.1\webapps\jetspeed\decorations\layout\purpleplanet\header.vm



6. As shown in the image above, underneath **<div class="header">** is the line ****

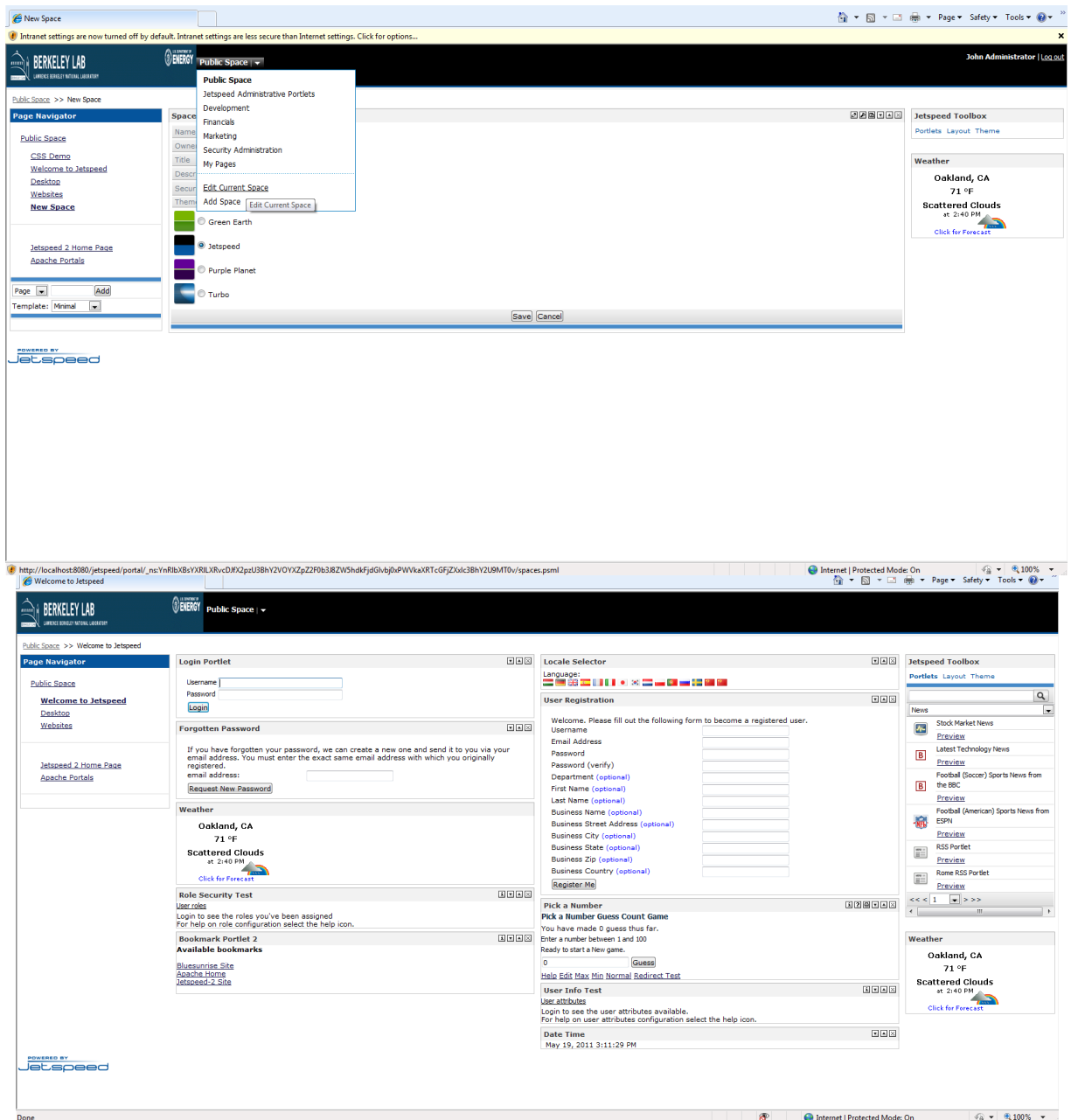
- a. This line refers to the logo being used in this file.
(Note: In the original file, the line was **<h1 class="logo">Jetspeed 2</h1>**)

7. To change this logo to the logo that was saved in step 4, delete the line below **<div class="header">** and replace it with the following:

- a. ****

8.

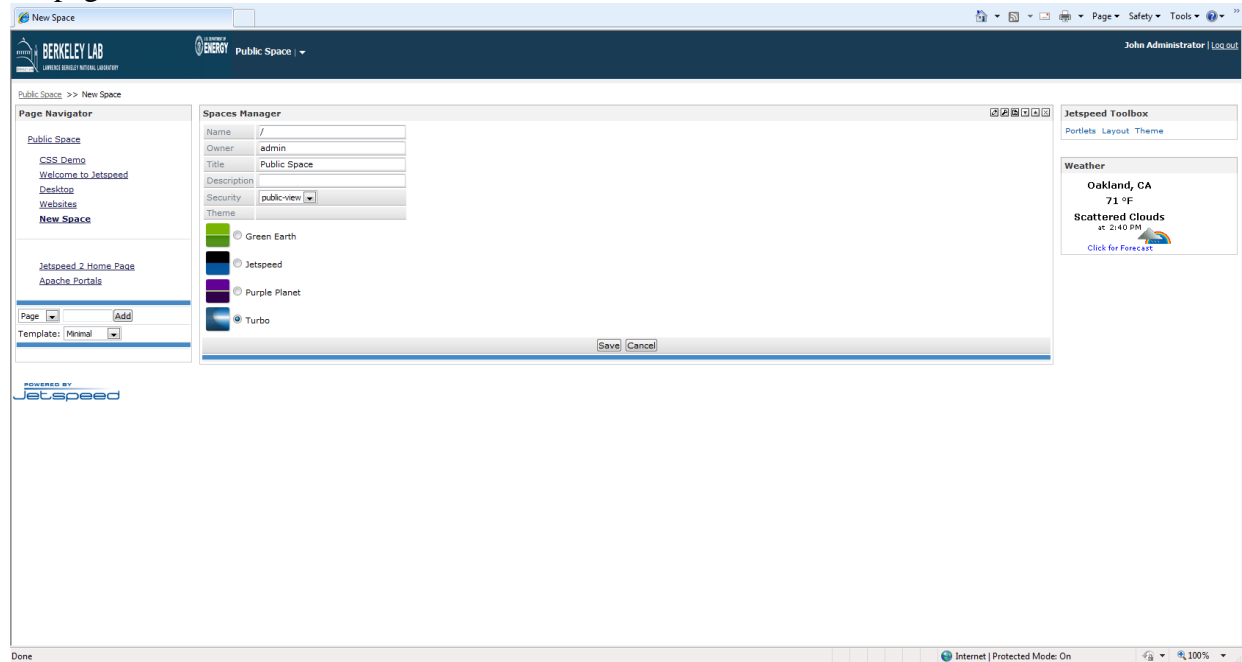
8. Your header will now be visible in Jetspeed as shown in the image below



11.7.3 How to change the header background color for the new header logo

- To set the background color of the header go the “Public Space” and select “Edit Current Space”

10. Then, select the background color from the options shown below
11. Then, repeat step 11 to set the header background color to the color you want for all the pages.



12 Deploying UDM Portlets

Download the package called udmUI.zip and unzip it.

12.1 High level instruction

1) Install a Java portlet container or application server such as Jetspeed-2 or Jboss.

2) Configure the application server to create a Database Connection Pool (DBCP) for connecting to your UDM database server. If you are using Jetspeed-2, use the instruction in this directory on how to set up DBCP. If you are using any other application server, follow that application server's documentation to set up the DBCP.

3) Start your application server. In case of Jetspeed-2, you need to run the script startup.sh (for liunux) or startup.bat (for Windows) located in the \$CATALINA_HOME/bin directory.

4) Deploy the web-app udmUI.war. In case of Jetspeed-2, you have copy the war file and paste it in the \$CATALINA_HOME/webapps/jetspeed/WEB-INF/deploy directory.

5) Then you can go to your portal homepage by typing `http://hostname:8080` in the address bar of your browser window. Note, if you have used the tomcatStartStop script, it automatically redirects port 8080 to port 80.

12.2 Database Connection Pool (DBCP)

Set up Database Connection Pool in your J2EE application server. Following is Jetspeed specific instruction.

1. Open file \$CATALINA_HOME/conf/context.xml

2. Add the following Resource item as a child element of the <Context> element.

```
<Resource name="jdbc/udmDS" auth="Container"
factory="org.apache.commons.dbcp.BasicDataSourceFactory"
type="javax.sql.DataSource" username="DB username" password="DB user password"
driverClassName="org.postgresql.Driver" url="jdbc:postgresql://hostname:5432/udm"
maxActive="100" maxIdle="30" maxWait="10000"/>
```

3. Modify the <Context> element, if needed, to read like the following:

```
<Context crossContext="true">
```

4. Ensure that the following jar files are present in the \$CATALINA_HOME/lib directory.

- a) commons-dbcp-1.4.jar
- b) commons-pool-1.6.jar
- c) PostgreSQL JDBC driver jar file

13 Text File Data

13.1 The command

The following command is issued in the script.

```
java -jar udmTextFile.jar db_hostname db_username db_password pc_hostname  
pc_username pc_password DataSourceProtocolID "America/Los_Angeles" run_type  
backfill_starttime backfill_endtime
```

run_type could be either “realtime” or “backfill” and if backfill, then the following two inputs need to be provided.

13.2 Parsing algorithm

```
/**  
 * DataSourceDetail1 = number of lines to skip before the header line  
 * DataSourceDetail2 = timestamp_source ("system" OR not)  
 *  
 * If DataSourceDetail2 = "system"  
 *     To be implemented  
 *     DataSourceDetail3 = (filter_string, value) pair in a single row or in a single column  
 *         (= "singlerow" OR not)  
 *     DataSourceDetail4 = filter_string  
 *  
 * Else  
 *     DataSourceDetail3 = headers are in different rows (of the left most column)  
 *         or  
 *         headers are in different columns (of the top row)  
 *         (= "rowbased" OR not)  
 *  
 *     If DataSourceDetail3 = "rowbased"  
 *         To be implemented  
 *  
 *     Else  
 *         DataSourceDetail4 = ";" separated timestamp_column_names  
 *             First one(s) is/are base, the last one is offset  
 *             If there is only one base, timestamp will be  
 *                 calculated as, (base + offset * factor)  
 *             If there are multiple base columns such as  
 *                 one for year, a second one for the month etc,  
 *                 first the resultant base will be created by
```

* concatenating the components specified by
 * the ";" separated elements, and then
 * the (base + offset * factor) will be performed
 * If there is no offset, the last element in the
 * ";" separated elements has to be "none"
 *

* DataSourceDetail5 = ";" separated timestamp_format and offset format
 * Allowed offset formats are ww (week), dd (day),
 * HH (hour), mm (minute) and ss (second). The offset
 * number could be integer or float. Essentially this
 * number is multiplied with the appropriate factor
 * to obtain the ms of time it represents. For example,
 * if the format is HH, the number will be multiplied
 * by (3600 * 1000)
 * If there is no offset, that is, if the last element
 * in DataSourceDetail4 is "none", then
 * BaseElementCount = number of elements for base; and
 * ONLY the first BaseElementCount number of elements
 * in DataSourceDetail5 are considered
 *

* DataSourceDetail6 = semicolon separated
 * (point_filter_column_name=point_filter) pair.
 *

* DataSourceDetail6 should be null
 *

* For Spreadsheet format, where there is
 * individual value_column for respective points
 *

* Calculation Steps:
 * -----
 * Remove all spaces
 * Split by ";" to get the pairs
 * Split every pair by "="
 *

* DataSourceDetail7 = semicolon separated (value_column_name=additional_seconds
 * from the timestamp in the timestamp column) pair.
 *

* value_column_name=0
 *

* For Spreadsheet format, where there is

```

*          individual value_column for respective points
*
*          OR
*
*          For no additional seconds, where value_column
*          contains just the values such as the following:
*
*          point_description  timestamp  value
*          -----  -
*          point1            t11        v11
*          point2            t21        v21
*          point1            t12        v12
*          point3            t31        v31
*          point2            t22        v22
*          ...               ...        ...
*          ...               ...        ...
*
*
*          DataSourceDetail8 = ";" separated InsertTime_column_names (base first, offset next)
*          DataSourceDetail9 = ";" separated InsertTime_format and offset format
*
*
* Time format
* -----
* Letter  Date or Time Component  Presentation  Examples
* -----  -----
* G      Era designator          Text          AD
* y      Year                    Year           1996; 96
* M      Month in year           Month         July; Jul; 07
* w      Week in year            Number        27
* W      Week in month           Number        2
* D      Day in year             Number        189
* d      Day in month            Number        10
* F      Day of week in month     Number        2
* E      Day in week             Text          Tuesday; Tue
* a      Am/pm marker            Text          PM
* H      Hour in day (0-23)       Number        0
* k      Hour in day (1-24)       Number        24
* K      Hour in am/pm (0-11)    Number        0
* h      Hour in am/pm (1-12)    Number        12
* m      Minute in hour          Number        30
* s      Second in minute        Number        55
* S      Millisecond             Number        978

```

* z	Time zone	General time	Pacific Standard Time; PST; GMT-08:00
*		zone	
* Z	Time zone	RFC 822 time	-0800
*		zone	
*/			

14 XML Data

14.1 The command

```
java -jar udmXMLFile.jar db_hostname db_username db_password pc_hostname  
pc_username pc_password DataSourceProtocolID timezone realtime
```

timezone should be the timezone where the ProtocolCommunicator is running. An example is "America/Los_Angeles"

14.2 Parsing algorithm

```
/**
```

```
 * DataSourceDetail1 = "<" separated node names of the nodes in the tree
```

```
 *           containing the node carrying the timestamp (up/down). Each part of the
```

```
 *           timestamp will have to be separated by ";". We are using "<" because it
```

```
 *           is an illegal character to be a part of an XML element name. ( base
```

```
 *           first, offset next). To indicate the content is an attribute of the node,
```

```
 *           specify :attr=AttributeName, and to indicate that the content is the value
```

```
 *           of the node, nothing extra needs to be specified. The program first splits
```

```
 *           DataSourceDetail1 with ";" and then splits each resulting component with
```

```
 *           ":attr="
```

```
 *           If the timestamp/value pair is specified as the following:
```

```
 *           <abcd>
```

```
 *           <Temperature>
```

```
 *           <Value Timestamp="2011/06/20 11:23:00" Unit="degF">60.9</Value>
```

```
 *           <Value Timestamp="2011/06/20 11:33:00" Unit="degF">61.4</Value>
```


* </Temperature>

* </abcd>

*

* For the examples given below the description of DataSourceDetail7,

*

* 1)

parent<parent<parent<validDate;parent<parent<validTime;child<DataSourceDetail7

*

* 2) parent<parent<parent<parent<validDate;parent<parent<parent<validTime;

* parent<DataSourceDetail7

*

* 3) parent<parent<validDate;parent<validTime

*

* All "parent" and "child" are relative to DataSourceDetail6

*

* DataSourceDetail2 = ";" separated time format for interpreting the value from the file

* DataSourceDetail3 = ";" separated time format which will

* finally be extracted from the values specified in the nodes of

* DataSourceDetail1. The number of entries in DataSourceDetail1,

* DataSourceDetail2 and DataSourceDetail3 must be the same and mapping to

* each other. If the format to be specified in DataSourceDetail3 is to be

- * the same as that in DataSourceDetail2, it should be specified as the
- * string "same".
- * DataSourceDetail6 = node corresponding to the quantity
- * There could be multiple instances in a single file corresponding to
- * different timestamps.
- * DataSourceDetail7 = ";" separated (value_node_name=additional_seconds) pairs.
- * The value_node_name is a node
- * name that has a sense of time associated with it (not as a value, but in
- * an implicit fashion) and that contains the value in its subtree. For
- * example, consider an hour to be divided into 4 intervals like the
- * following,
- *
- * 1) -----
- * <abcd>
- * <validDate> Oct 13 </validDate>
- * <period>
- * <validTime>00</validTime>
- * <temperature>
- * <INTERVAL01> 65.4 </INTERVAL01>
- * <INTERVAL02> 67.4 </INTERVAL02>
- * <INTERVAL03> 65.4 </INTERVAL03>

- * <INTERVAL04> 67.4 </INTERVAL04>
- * </temperature>
- * </period>
- * </abcd>
- *
- * 2) -----
- * <abcd>
- * <validDate> Oct 13 </validDate>
- * <period>
- * <validTime>00</validTime>
- * <INTERVAL01>
- * <temperature> 65.4 </temperature>
- * </INTERVAL01>
- * <INTERVAL02>
- * <temperature> 67.4 </temperature>
- * </INTERVAL02>
- * <INTERVAL03>
- * <temperature> 65.4 </temperature>
- * </INTERVAL03>
- * <INTERVAL04>

* <temperature> 67.4 </temperature>

* </INTERVAL04>

* </period>

* </abcd>

*

* 3) *****

* <abcd>

* <validDate> Oct 13 </validDate>

* <period>

* <validTime>00</validTime>

* <temperature> 67.4 </temperature>

* </period>

* </abcd>

*

* For both Case1 and Case2, DataSourceDetail7 should be

* INTERVAL01=900;INTERVAL02=1800;INTERVAL03=2700;INTERVAL04=3600;

*

* DataSourceDetail8 = "<" separated node names of the nodes in the tree

* containing the node carrying the forecast time (top down). Each part of

* the timestamp will have to be separated by ";". DataSourceDetail9 = ";

* separated InsertTime_format

$\ast/$

15 Modbus Data

15.1 Description

The script, `udmModbus`, allows the `udm java` application, `Modbus`, to run as a daemon².

Platform: Red Hat 4.1.2-50 (CentOS release 5.6)

15.2 Point details

<code>datasourceaddress1</code>	IP address/FQDN/hostname[:port]
<code>datasourceaddress2</code>	UnitID
<code>datasourceaddress3</code>	Start register
<code>datasourceaddress4</code>	Bit number (Only for discretes)
<code>datasourcedetail1</code>	Modbus Function Code
<code>datasourcedetail2</code>	Modbus data type
<code>datasourcedetail3</code>	<code>modulo_divisor</code> ;bits per modulo (in case of modulo)
<code>datasourcedetail4</code>	TotalBitCount
<code>datasourcedetail5</code>	A;B;C, where <code>final_value = A * (read_value ^ B) + C</code>

15.3 Installation

Copy `udmModbus` (which is stored in the `Modbus` directory) to the directory where the system's init scripts are stored (In this tutorial, this directory will be referred to as `/etc/init.d`). To verify that `udmModbus` has been copied into the `/etc/init.d` directory run the command `ls` in this directory. Once it has been verified that the script has been successfully copied into the `/etc/init.d` directory, run the command `chkconfig --add udmModbus`. To verify that `udmModbus` has been added to the run levels, run the command `chkconfig --list`.

15.4 Operation

To start, stop or check the status of `Modbus` in any directory use the following commands:

```
sudo /etc/init.d/udmModbus start
```

```
sudo /etc/init.d/udmModbus stop
```

```
sudo /etc/init.d/udmModbus status
```

² Daemon: A background process that handles requests for services such as print spooling and is dormant when not required (<http://www.wordreference.com/definition/demon>).

15.5 Troubleshooting

Common Errors:

1. Modbus does not start
2. Unable to find jar file error
3. The following set of java errors:

```
java.io.FileNotFoundException: null/etc/jdbc.conf (No such file or
directory)
    at java.io.FileInputStream.open(Native Method)
    at java.io.FileInputStream.<init>(FileInputStream.java:137)
    at java.io.FileReader.<init>(FileReader.java:72)
    at
gov.lbl.udm.DatabaseUtility.getJDBCstring(DatabaseUtility.java:90)
    at gov.lbl.udm.middleware.modbus.Main.main(Main.java:105)
java.io.FileNotFoundException: null/etc/DatabasePort.conf (No such file
or directory)
    at java.io.FileInputStream.open(Native Method)
    at java.io.FileInputStream.<init>(FileInputStream.java:137)
    at java.io.FileReader.<init>(FileReader.java:72)
    at
gov.lbl.udm.DatabaseUtility.getDatabasePort(DatabaseUtility.java:122)
    at gov.lbl.udm.middleware.modbus.Main.main(Main.java:108)
java.sql.SQLException: No suitable driver found for uf.lbl.gov:/udm
    at java.sql.DriverManager.getConnection(DriverManager.java:640)
    at java.sql.DriverManager.getConnection(DriverManager.java:200)
    at
gov.lbl.udm.DatabaseUtility.connectToDatabase(DatabaseUtility.java:18
6)
    at gov.lbl.udm.middleware.modbus.Main.main(Main.java:110)
Exception in thread "main" java.lang.NullPointerException
    at gov.lbl.udm.middleware.modbus.Main.main(Main.java:118)
```

Cause: Inside udmModbus, the current directory is not being changed to the Modbus directory.

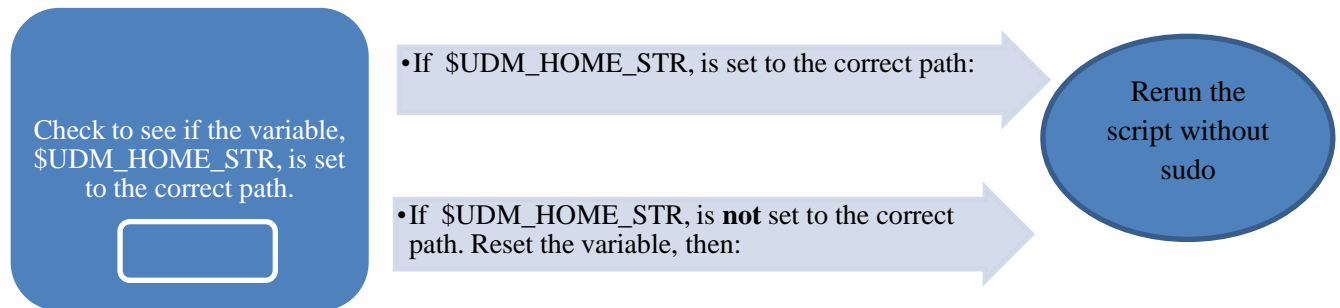
In order for Modbus to start via the script, the directory must be changed to the Modbus directory so that the script has access to the necessary files stored in the Modbus directory

Solution: Go to the script, create_udm_home.³ Inside this script, check to see if the variable UDM_HOME is being exported. An example of this command is the following:

```
export UDM_HOME=/data/software/udm
```

If that command is not there or the variable, UDM_HOME, is set to the wrong path go to the script, udmSetupENV.sh.⁴

Once inside the script:



³ This is the script that is called after the initialization of variables inside udmModbus. It should be located in the directory where the udm zip directories are stored.

⁴ This script is located in the unzipped directory, udmDatabaseSetup

16 IPMI Data

16.1 Point details

datasourceaddress1	IP address/FQDN/hostname[:port]
datasourcedetail1	Unique Identifier of the line containing the physical quantity of the point
datasourcedetail2	String w.r.t. which the 1st split will happen
datasourcedetail3	String w.r.t. which the 2nd split will happen
datasourcedetail5	A;B;C, where $\text{final_value} = A * (\text{read_value} ^ B) + C$

17 RecentSweeper

In realtime mode data is sent to a table called RecentData. In case of forecast data, it is sent to a table called RecentForecast. These tables act like a buffer zone for the data. When in realtime mode, another udm program needs to be run to sweep these "Recent" tables and send the data contained therein to the respective DataTables for the individual Points. This sweeper program is distributed as the package called udmRecentSweeper.zip. Download it and run it, and you will see your data in the DataTables. Once you start, this program will run continuously.

In contrary, in the backfill mode, data is sent directly to the DataTables because each individual (time,value) tuple is checked against certain timestamps (basically the StartTime of the dataTable it would go into). If needed, new DataTables are created for data that are older than the existing DataTables should contain. Overall, this process is more rigorous than the realtime mode. Consequently, the performance is much better in realtime mode and it is suitable for most situations. But if there is any possibility that the timestamp you are entering could be from a time earlier than the Point was created, the backfill mode needs to be used (in certain situations, you can get by, but that is what is recommended).

The following is the command that is executed inside the script file.

```
java -jar udmRecentSweeper.jar db_hostname db_username db_password  
[pc_hostname pc_username pc_password]
```

db_hostname = the hostname, FQDN or the IP address of the database server

db_username = database username - this should have enough access to create new database

db_password = password corresponding to db_username

If the ProtocolCommunicators (pc) are running on hosts separate from the ones where the UDM database is hosted:

pc_hostname = the hostname, FQDN or the IP address of the pc host

pc_username = username for the database running on pc - this should have enough access to delete rows

pc_password = password corresponding to pc_username

18 UDM and BCVTB

The UDM-BCVTB bridge is a collection of Ptolemy II Actors and they are distributed in source code form in the package called udmPtolemyBCVTB.zip.

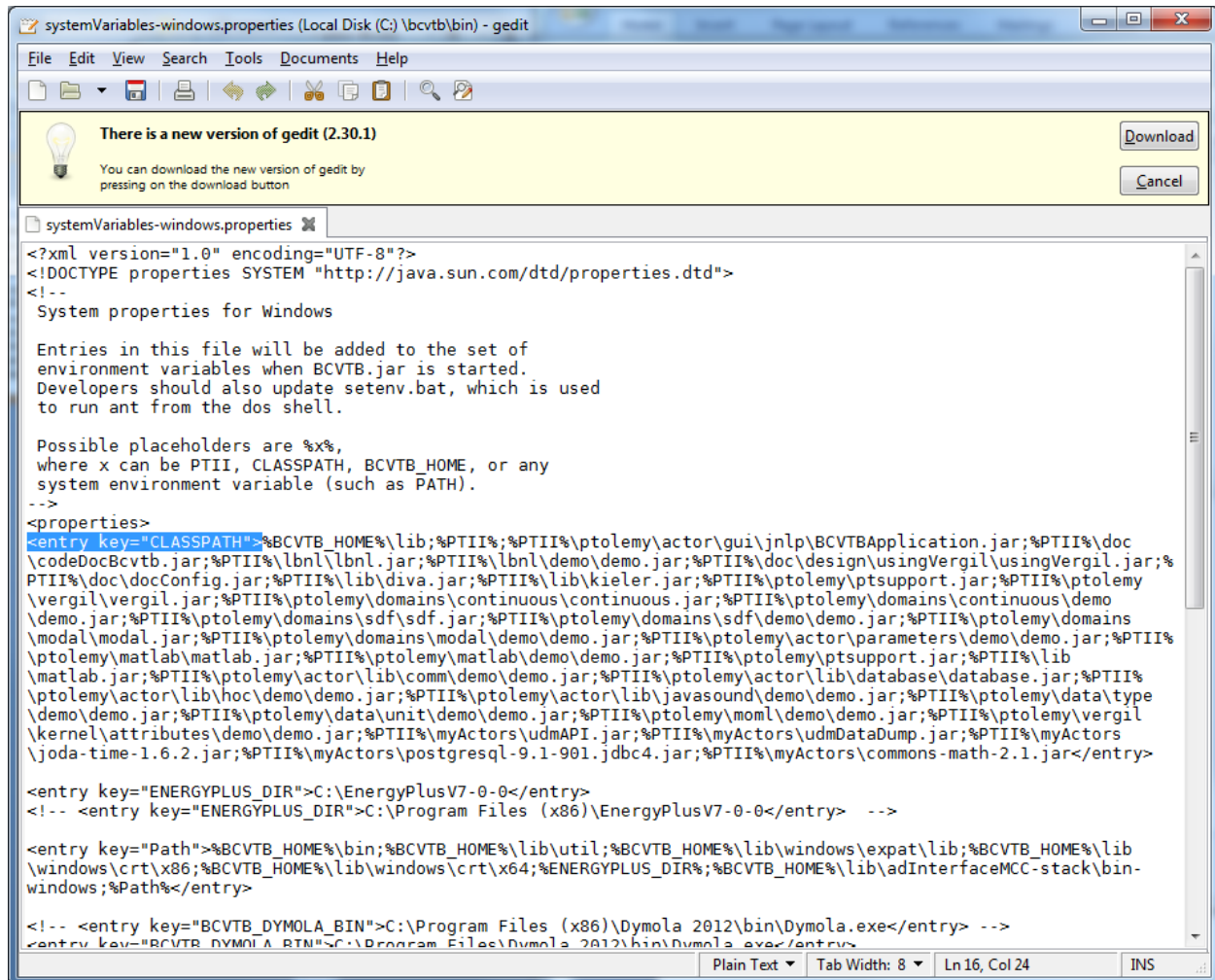
18.1 Libraries to be downloaded

1. Joda-time (<http://joda-time.sourceforge.net/>)
2. Apache commons math (http://commons.apache.org/math/download_math.cgi)
3. PostgreSQL JDBC4 driver (<http://jdbc.postgresql.org/download.html>)
4. udmAPI.zip
5. udmDataDump.zip

18.2 Setup work

1. The downloaded libraries need to be extracted and the resulting *.jar files need to be placed in the BCVTB_HOME/lib/ptII/myActors directory.

2. Edit the CLASSPATH entry in file BCVTB_HOME\bin\systemVariables-linux.properties. See the attached screenshot.



3. Add the following entries at the end of the already existing value for CLASSPATH.

For Windows:

%PTII%\myActors\udmAPI.jar;%PTII%\myActors\udmDataDump.jar;%PTII%\myActors\joda-time-1.6.2.jar;%PTII%\myActors\postgresql-9.1-901.jdbc4.jar;%PTII%\myActors\commons-math-2.1.jar

For Linux:

%PTII%\myActors\udmAPI.jar;%PTII%\myActors\udmDataDump.jar;%PTII%\myActors\joda-time-1.6.2.jar;%PTII%\myActors\postgresql-9.1-901.jdbc4.jar;%PTII%\myActors\commons-math-2.1.jar

4. Extract the udmPtolemyBCVTB.zip package and copy the content of the directory into BCVTB_HOME/lib/ptII/myActors directory.
5. Run setDevelopmentEnvironment in the bin folder in BCVTB_HOME

6. Run the command `ant clean all` to build and compile the files (You need to have a C/C++ compiler already installed on your computer.)
7. Run BCVTB and check if you have the actors under the MyActors folder

18.3 Using the combined power of UDM and BCVTB

18.3.1 BACnet interface

The BACnet module in the BCVTB, described in [14], interfaces with the Siemens BACnet server in order to collect the building performance data. The procedure for using the BACnet interface involves the following steps:

18.3.1.1 Information gathering

Collect the BACnet server/device information from the vendor, including device instance, object type and mapping of all the data points to relevant object type instances. The device instance can be verified by running `globalwi` in the `BACnet-stack/bin-linux` directory, as illustrated in Figure 17. As can be seen from **Error! Reference source not found.**, there are five devices (Siemens MEC controllers), with instance numbers 7150 to 7154, in the Siemens EMS.

```
[xpang@estcp-site1 bin-linux]$ ls
bacrp  bacwp  globalwi  readobjlist
[xpang@estcp-site1 bin-linux]$ ./globalwi
Received I-Am Request from 7154, MAC = 172.16.10.10.186.192
Received I-Am Request from 7153, MAC = 172.16.10.10.186.192
Received I-Am Request from 7152, MAC = 172.16.10.10.186.192
Received I-Am Request from 7151, MAC = 172.16.10.10.186.192
Received I-Am Request from 7150, MAC = 172.16.10.10.186.192
7154
7153
7152
7151
7150
[xpang@estcp-site1 bin-linux]$
```

Figure 18-1: Results of running `globalwi`

18.3.1.2 Configuration file

Develop an xml configuration file. Three types of objects need to be specified in the xml configuration file:

- BACnet. This is the root of the xml configuration file, every file has to start with `<BACnet>` and end with `</BACnet>`. Only contents specified between these delimiters will be recognized by BACnetreader.
- ObjectType. This is used to specify BACnet objects, including device objects and non-device objects. Non-device objects are attached to a device object, therefore, non-device objects are specified at the child level of device objects, although they use the same name “ObjectType”. For device objects, the name attribute should be “DeviceObjectType”, the instance attribute should be the device instance number. For non-device objects, the name attribute should be the

name of the object. For example, for “AnalogInputObjectType”, the instance attribute should be the object instance number.

- **PropertyIdentifier.** This is used to specify the BACnet properties that the user wants to query from the BACnet server/device. They can be properties of both a device object and a non-device object. They should be at the child level of corresponding objects. The name attribute should be the name of the property.

Figure 18-2 shows the configuration file used in the Great Lakes installation.

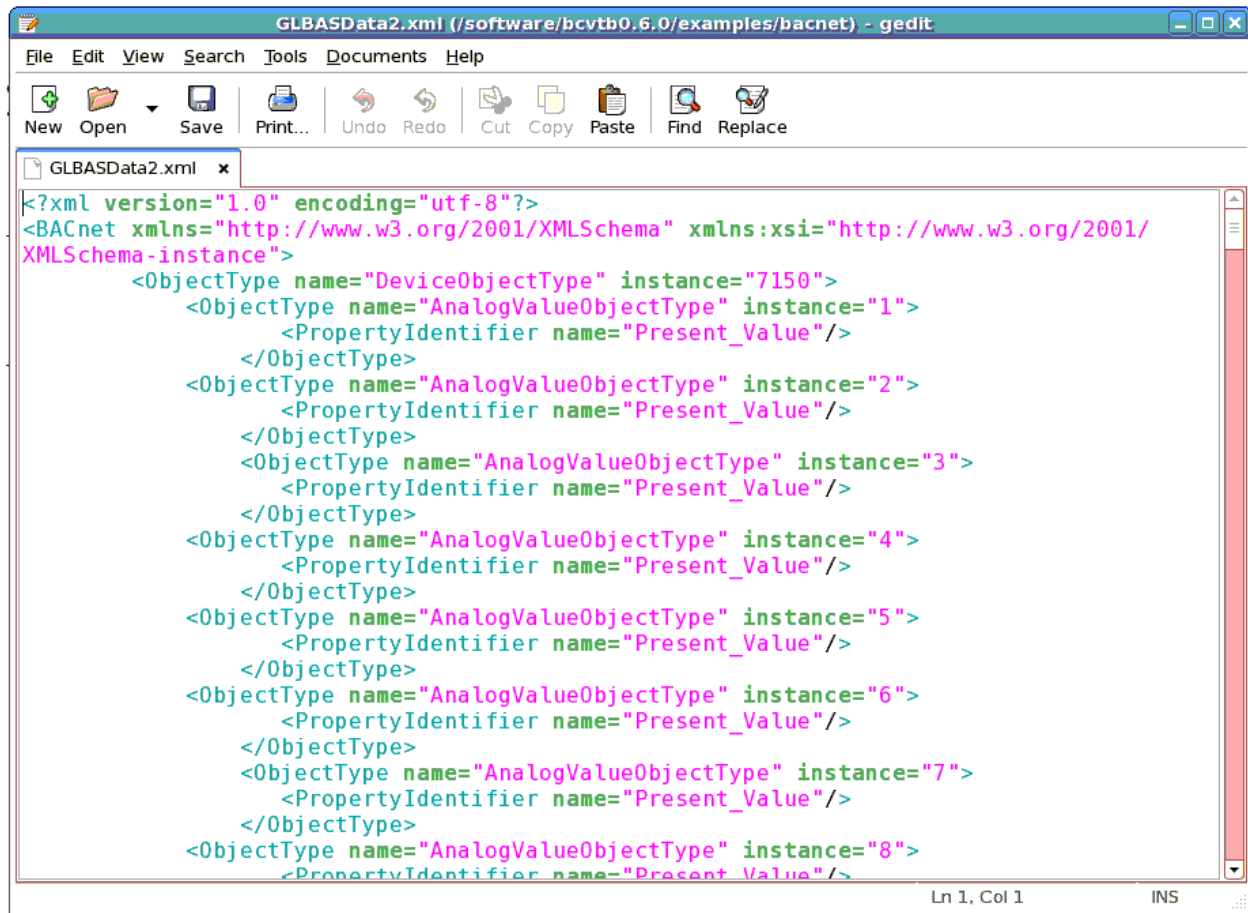


Figure 18-2: BACNet configuration file

18.3.1.3 Ptolemy model

Develop a Ptolemy model. An example model is shown in Figure 18-3. By double clicking the BACnet module, a configuration window will pop up. The xml file configured in step 2 needs to be specified here. The sampling interval can be specified by double clicking the SDF director.

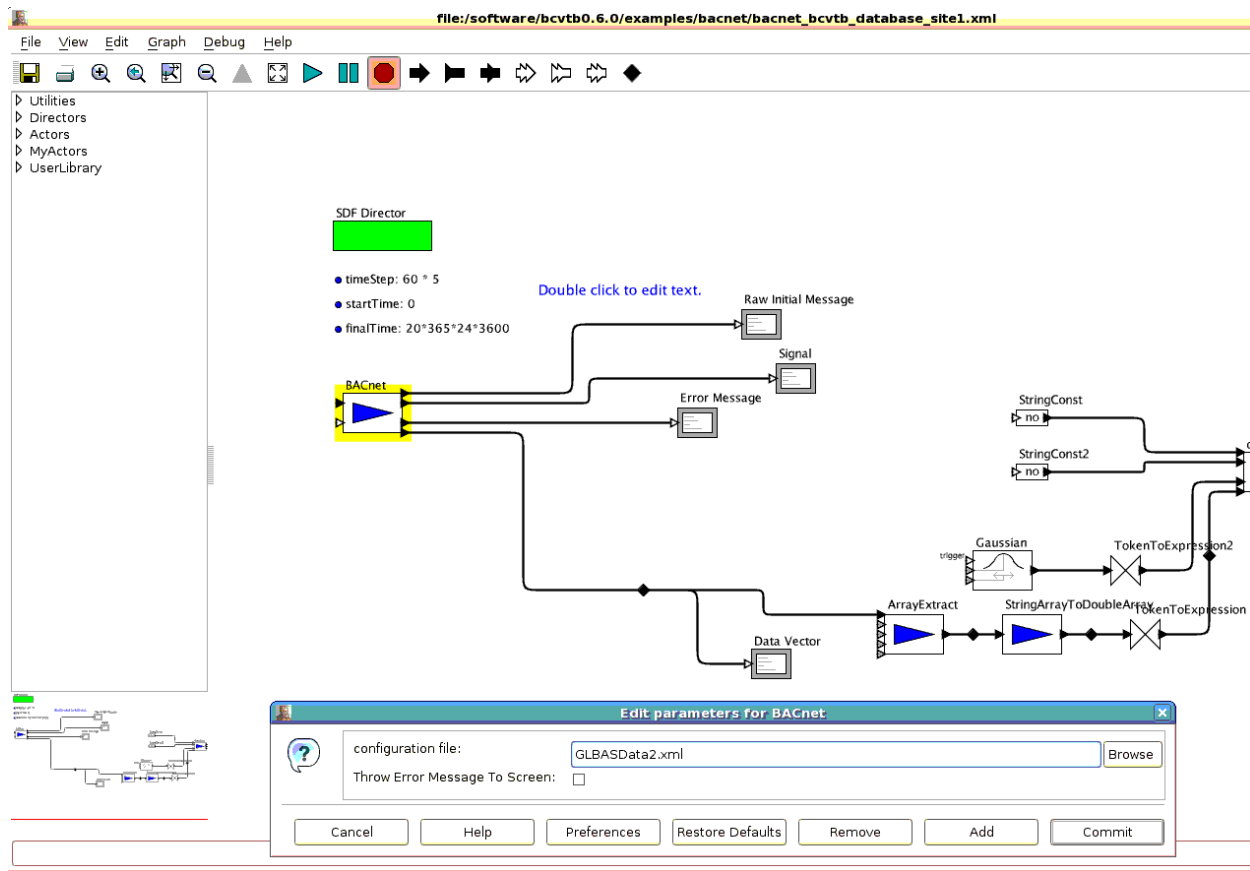


Figure 18-3: BACnet model presented in the Ptolemy II UI

18.3.2 BCVTB-UDM integration

The integration involves the following steps:

- 1) Set up UDM.
- 2) At this point, it is assumed that you have already built BCVTB to include the Actors distributed as part of UDM under the udmPtolemyBCVTB package.
- 3) Build one Ptolemy model for sending BACNet data from BCVTB to UDM. And build another Ptolemy model for the realtime EnergyPlus. All the relevant Actors will be available under the MyActor directory.
 - The actor called udmManager has to be there in any model that tries to connect to UDM.
 - The actor called writeToUDM is used to send data to UDM. It takes a vector of double data type as input and based on the configuration specified, sends the data to the database.
 - The actor called readTWMeanFromUDM fetches time-weighted values from the database. For example, if the BACNet data is being collected and stored once every 5 minutes and the E+ timestep is 15 minutes (meaning E+ needs input data once every 15 minutes), one may want to send to E+ the time-weighted mean of each variable over the last 15 minutes. That is where this Actor becomes useful. The output of this Actor is a vector of double data type.

18.3.3 Real time EnergyPlus in the BCVTB

18.3.3.1 External Interface

In EnergyPlus, the External Interface objects are used to exchange data between EnergyPlus and the BCVTB. The objects can map to three EnergyPlus input objects called ExternalInterface:Schedule, ExternalInterface:Actuator and ExternalInterface:Variable. The ExternalInterface:Actuator was used to implement real-time EnergyPlus at Great Lakes. This object behaves identically to EnergyManagementSystem:Actuator, with the following exceptions:

- Its value is assigned by the external interface.
- Its value is fixed during the zone time step because this is the synchronization time step for the external interface.

18.3.3.2 Configuration

Configuring the data exchange involves the following three steps:

Create an EnergyPlus idf file

Figure 18-4 shows how to set up the part of an EnergyPlus input file that specifies the name of the External Interface using IDF Editor.

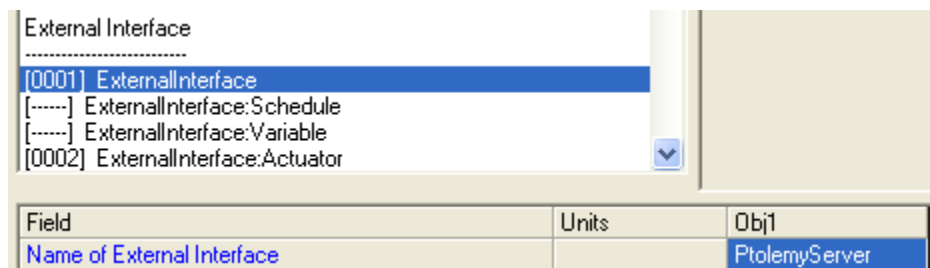


Figure 18-4: Specifying the External Interface in the EnergyPlus IDF Editor

Figure 18-5 shows how to declare actuators that update the outdoor dry bulb and relative humidity values in EnergyPlus. It is worth noting that actuators to update the weather data are only available in EnergyPlus Version 6 and later.

External Interface

[0001] ExternalInterface
[.....] ExternalInterface:Schedule
[.....] ExternalInterface:Variable
[0002] ExternalInterface:Actuator

Field	Units	Obj1	Obj2
Name		OADryBulb	OARH
Actuated Component Unique Name		Environment	Environment
Actuated Component Type		Weather Data	Weather Data
Actuated Component Control Type		Outdoor Dry Bulb	Outdoor Relative Humidity
Optional Initial Value		30	10

Figure 18-5: EnergyPlus setup interface for external real time weather information

If the optional field that specifies the initial value is unspecified, then the actuator will only be used during the real time operation, but not during the warm-up and the system sizing. Since actuators always overwrite other objects (such as schedules), all these objects have values that are defined during warm-up and system sizing, even if no initial value is specified in the ExternalInterface:Actuator.

18.3.3.3 Create an xml file

It is necessary to specify the order of the elements in the signal vector that is exchanged between EnergyPlus and the BCVTB. This information is specified in the file variables.cfg. The file variables.cfg needs to be in the same directory as the EnergyPlus idf file. The file has the following form:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE BCVTB-variables SYSTEM "variables.dtd">
<BCVTB-variables>
  <variable source="EnergyPlus">
    <EnergyPlus name="Space1-1" type="Zone/Sys Sensible Cooling Rate"/>
  </variable>
  <variable source="EnergyPlus">
    <EnergyPlus name="Space2-1" type="Zone/Sys Sensible Cooling Rate"/>
  </variable>
  <variable source="Ptolemy">
    <EnergyPlus actuator="OADryBulb" />
  </variable>
  <variable source="Ptolemy">
    <EnergyPlus actuator="OARH" />
  </variable>
</BCVTB-variables>
```

The <variable source="Ptolemy"> entry specifies the element written from the BCVTB to EnergyPlus. The <variable source="EnergyPlus"> entry specifies the element computed by EnergyPlus and sent to the BCVTB.

18.3.3.4 Create a Ptolemy model

A Ptolemy model is needed to start EnergyPlus from the BCVTB. An example model is shown in Figure 18-6.

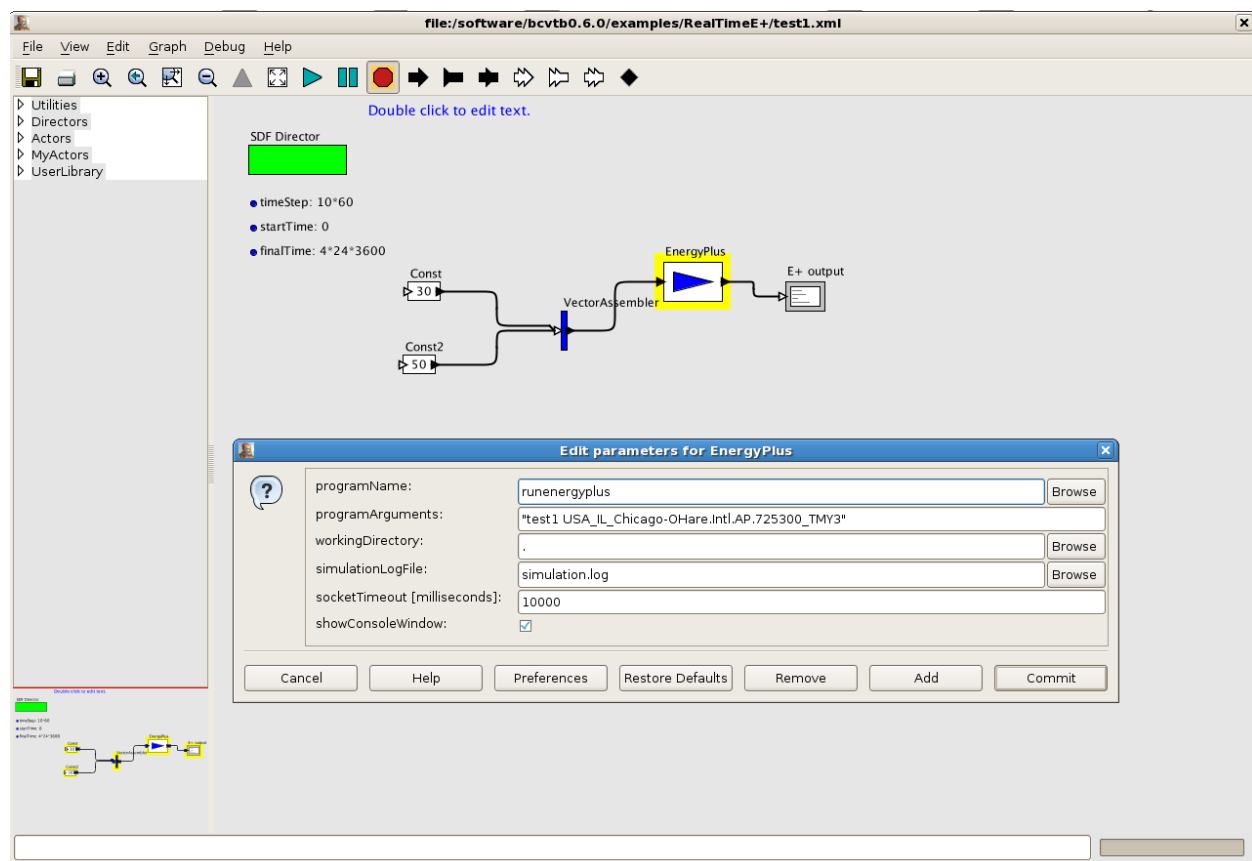


Figure 18-6: Ptolemy interface to setup EnergyPlus

Double clicking the EnergyPlus module, causes a configuration window to pop up. In the “programName” field, the default entry “runenergyplus” does not need to change. The “programArguments” field is about the only one that needs to be edited. In this example, the EnergyPlus idf file name is specified to be “test1”, as in the example, and the weather file name is specified to be “USA_IL_Chicago-OHare.Intl.AP.725300_TMY3” as in the example. The working directory is the current directory and the console output is written to the file simulation.log. If EnergyPlus does not communicate with the BCVTB within 10 seconds, the BCVTB will terminate the connection, which is specified in the “socketTimeout” field.

The sampling interval can be specified by double clicking the SDF director. In this example, the time step is 10 minutes and the simulation period is four days. The same time step needs to be specified in the idf file.

18.3.4 BCVTB setup

18.3.4.1 Start-up

The following command starts the BCVTB module containing the BACnet and the database connector components. This should be started only after the database is started.

```
$java -jar BCVTB.jar -console /software/bcvtb0.5/examples/bacnet/bacnet_bcvtb_database_site1.xml
```

A shortcut script can be created to perform this task.

In order to change the points list, the following steps have to be followed

- Make the appropriate changes in the csv file that holds the input points list for either BACNet or E+ input or E+ output
- Generate a new xml file from the above csv files. This is the xml file that holds the BACnet points list (fileA.xml) or the E+ points list, to be used by BCVTB.
- If the name of the xml file was changed, insert the correct xml filename (fileA.xml) into field "" in the xml file for the Ptolemy II model
- If the name of the csv file was changed, insert the correct csv filenames in the input fields for the Actors writeToUDM and readTWMeanFromUDM

18.3.4.2 Upgrading BCVTB

Copy the bcvtb.version_number directory of the new version over to the production machine under the directory of your choice. Then the following directories on the production machine have to be copied from the earlier installation directory to the new installation directory.

bcvtb.version_number/lib/ptII/myActors

bcvtb.version_number/lib/ptII/bacnet

bcvtb.version_number/lib/bacnet-stack

19 Using UDM API

The package `udmAPI.zip` contains the `udmAPI.jar` library as well as the javadoc documentation of the API. The essential steps are:

1. Establishing connection to the database
2. Querying the database to obtain the list of Points (both input and output) with which the calculation will be performed
3. Querying the database to pull data for the input Points
4. Performing the calculation
5. Push data (the results of the calculation) to the database for the output Points

The following code is from the package `udmDataServingDemo.zip` which is meant show developers how a 3rd party application can be built using the UDM API.

```
package gov.lbl.udm.dataserving.sample;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.sql.Connection;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;


import gov.lbl.udm.DatabaseUtility;

import gov.lbl.udm.metadata.Point;

import gov.lbl.udm.metadata.PointUtility;

import gov.lbl.udm.middleware.ProtocolCommunicator;
```

```

import gov.lbl.udm.timeseries.DatabaseQuery;

import gov.lbl.udm.timeseries.ExcessInterpolationTimestampSourceException;

import gov.lbl.udm.timeseries.IllegalTimeFormatException;

import gov.lbl.udm.timeseries.NoInterpolationTimestamp;

import gov.lbl.udm.timeseries.NoInterpolationTimestampSourceException;

import gov.lbl.udm.timeseries.NoSelectedPointException;

import gov.lbl.udm.timeseries.PointData;


public class Main {

    /**
     * @param args
     * @throws IllegalTimeFormatException
     */
    public static void main(String[] args) throws IllegalTimeFormatException
    {

        // Inputs specific to this sample class

        String database_hostname = args[0];

        String username = args[1];

        String password = args[2];

        String Start_Time = args[3]; // this is required for reading from the

```

```

// database

String End_Time = args[4]; // this is required for reading from the

// database

String run_type = args[5]; // this is required for writing to the

// database

// Valid values of run_type are "realtime" "backfill_insert"

// "backfill_upsert"

// and "backfill_replace"


String database_to_connect_to = "udm";

if ( args.length > 6 ) {

    database_to_connect_to = args[6];

}


// use of the API starts

DatabaseUtility DBUtil_Instance = new DatabaseUtility();

String udm_root_directory_string = DBUtil_Instance

.getUDM_root_directory_string();


String database_specific_jdbc_string = DBUtil_Instance

```

```

        .getJDBCstring(udm_root_directory_string);

String database_port = DBUtil_Instance

        .getDatabasePort(udm_root_directory_string);

Connection con = DBUtil_Instance.connectToDatabase(database_hostname,

        database_port, database_to_connect_to, username, password,

        database_specific_jdbc_string);

// use of the API ends

/*

* Part 1: Load the list of relevant Points as Java objects

*

* A) This part queries the point database based on pre-specified
unique

* point identifiers (at this point, possibly stored in a file, but

* later on stored in the database)

*

* B) This step can be performed i) only at the beginning of the

* continuously running iterative process (and cached for the rest of

* its life) or ii) intermittently (such as once every day) or iii)
at

* the beginning of each iteration.

```

```

*

* C) Typically one would like to maintain two lists - the first list
* containing points for which data is to be read from the database
and
* the second one containing points for which data is to be written
to
* the database

*/

// For this example, these values have been hard-coded. In a
production

// code, they should come from outside of the code, such as a file or
// a script file encompassing the java program.

File InputPoint_file = new File("InputPoint.csv");

// The following code assumes that this file has a single
// header line
// and the header line contains the actual database field names

// The following column counts need not be hard-coded if the .csv
files
// are well-formed.

// They can be obtained by parsing the file.

```



```

int InputPoint_file_ColumnCount = 3;

int i = 0;

int j = 0;

List<String[]> InputPointList = loadPointDetailsFromFile(
    InputPoint_file, InputPoint_file_ColumnCount);

// Note: The first String Array of the above List contains the header
// fields

int InputPointsCount = InputPointList.size() - 1;

String[][][] InputPoints_header_value_pair_list = new
String[InputPointsCount][InputPoint_file_ColumnCount][2];

for (i = 0; i < InputPointsCount; i++) {

    for (j = 0; j < InputPoint_file_ColumnCount; j++) {

        InputPoints_header_value_pair_list[i][j][0] = InputPointList
            .get(0)[j];

        InputPoints_header_value_pair_list[i][j][1] = InputPointList
            .get(i + 1)[j];

    }
}

```

```

    }

    // Use of the API starts

    PointUtility Query_PointUtility = new PointUtility();

    List<Point> InputPoints = Query_PointUtility.getPredefinedPointList(
        con, InputPoints_header_value_pair_list);

    // Use of the API ends

    // Part 2: Read data from the database

    // For this example, these values have been hard-coded. In a
production
    // code, they should come from outside of the code, such as a file or
    // a script file encompassing the java program or some other
    // programmatic source.

    String start_BoundaryType = ">=";

    String end_BoundaryType = "<=";

    Point interpolation_timestamp_source_Point = null;

    long time_interval = 0;

    // Use of the API starts

    long start_time = DBUtil_Instance.getLongOutputTime(Start_Time, "/",
2);

    long end_time = DBUtil_Instance.getLongOutputTime(End_Time, "/", 2);

```

```

List<PointData> rawInputData = new ArrayList<PointData>();

List<List<Double>> interpolatedInputData = new
ArrayList<List<Double>>();

DatabaseQuery DBQInstance = new DatabaseQuery();

try {

    // Example of querying for raw stored data

    // This has not been used in this code

    rawInputData = DBQInstance.getDataRawIndividualPoints(con,
        InputPoints, start_time, end_time, start_BoundaryType,
        end_BoundaryType);

    // Example of querying for interpolated data
    *****

    // -----

    // Use the first point in the list as the source of timestamps
    // to which the interpolation will be performed

    // interpolation_timestamp_source_Point = InputPoints.get(0);

    // -----

    // OR set the time_interval to a non-zero value

```

```

        time_interval = 45;

        // -----

        // Setting both to non-zero non-null values will result in
exception

        // In this List of List<Double>, the first List contains
        // the timestamps, each of the other Lists contain data
        // for the respective Point as ordered in the InputPoints List.
        // Hence, interpolatedInputData.get(i).get(j) would mean
        // the jth datapoint for the ith Point and
        // interpolatedInputData.get(0).get(j) would mean the jth
        // timestamp, sorted in increasing order
        // The timestamp is UTC in millisecond

        interpolatedInputData = DBQInstance
            .getDataInterpolatedUnfilteredLongInputLongOutput(con,
                InputPoints, interpolation_timestamp_source_Point,
                start_time, end_time, time_interval,
                start_BoundaryType, end_BoundaryType);

    } catch (SQLException e) {

        // TODO Auto-generated catch block

        e.printStackTrace();
    }

```

```

    } catch (NoSelectedPointException e) {

        // TODO Auto-generated catch block

        e.printStackTrace();

    } catch (NoInterpolationTimestampSourceException e) {

        // TODO Auto-generated catch block

        e.printStackTrace();

    } catch (ExcessInterpolationTimestampSourceException e) {

        // TODO Auto-generated catch block

        e.printStackTrace();

    } catch (NoInterpolationTimestamp e) {

        // TODO Auto-generated catch block

        e.printStackTrace();

    }

}

// Use of the API ends

/*

    * Part 3: Perform calculations Intentionally, no UDM class was used

in    * this section. This was done in order to simulate a 3rd party

    * application that's oblivious of UDM

    */

```

```

int calc_output_count = 2;

int output_timestamp_count = interpolatedInputData.get(0).size();

double[][] FinalOutput = new double[calc_output_count +
1][output_timestamp_count];

for (j = 0; j < output_timestamp_count; j++) {

    FinalOutput[0][j] = interpolatedInputData.get(0).get(j); //
Timestamps

    // Calculate  $Q = m * cp * (T_{\text{return}} - T_{\text{supply}})$ 

    FinalOutput[1][j] = interpolatedInputData.get(2).get(j)

    * 0.16

    * (interpolatedInputData.get(4).get(j) - interpolatedInputData

        .get(3).get(j));

    // Calculate % deviation =  $100 * (\text{calculated} - \text{measured}) /$ 
measured

    double measured_value = interpolatedInputData.get(1).get(j);

    if ( measured_value != 0 ) {

        FinalOutput[2][j] = 100

        * (FinalOutput[1][j] - measured_value) / measured_value;

```

```

    }
}

/*
 * Part 4: Write data back to the database
 *
 * A) This part queries the point database based on pre-specified
unique
 * point identifiers (at this point, possibly stored in a file, but
 * later on stored in the database)
 *
 * B) This step can be performed i) only at the beginning of the
 * continuously running iterative process (and cached for the rest of
 * its life) or ii) intermittently (such as once every day) or iii)
at
 * the beginning of each iteration.
 */

File OutputPoint_file = new File("OutputPoint.csv");

// The following code assumes that this file has a single
// header line
// and the header line contains the actual database field names

// The following column counts need not be hard-coded if the .csv

```

files

```
// are well-formed.

// They can be obtained by parsing the file.

int OutputPoint_file_ColumnCount = 3;

List<String[]> OutputPointList = loadPointDetailsFromFile(

    OutputPoint_file, OutputPoint_file_ColumnCount);

// Note: The first String Array of the above List contains the header

// fields

int OutputPointsCount = OutputPointList.size() - 1;


String[][][] OutputPoints_header_value_pair_list = new
String[OutputPointsCount][OutputPoint_file_ColumnCount][2];


for (i = 0; i < OutputPointsCount; i++) {

    for (j = 0; j < OutputPoint_file_ColumnCount; j++) {

        OutputPoints_header_value_pair_list[i][j][0] =
OutputPointList

            .get(0)[j];

        OutputPoints_header_value_pair_list[i][0][1] =
OutputPointList

            .get(i + 1)[j];
```



```

    }
}

// Use of the API starts *****

PointUtility Write_PointUtility = new PointUtility();

List<Point> OutputPoints = Write_PointUtility.getPredefinedPointList(
    con, InputPoints_header_value_pair_list);

// Assign values to the Points and send the Points through the data
// compression cycle

// Note: right now the data compression algorithm is not implemented.

// Data compression is turned on/off in the database in Point
// configuration

// Therefore it is advisable to keep the data compression code in
here

// that way data compression on/off can be controlled just by
// changing the configuration, as opposed to making changes to the
// source code

// of the application.

for (i = 0; i < OutputPointsCount; i++) {

    for (j = 0; j < FinalOutput[0].length; j++) {

```

```

data        // Note: the timestamp is UTC millisecond and a JAVA long

            // type

            OutputPoints.get(i).UnCompressedTimestamp

            .add((long) FinalOutput[0][j]);

            // Note: the value is a JAVA double data type

            OutputPoints.get(i).UnCompressedValue

            .add(FinalOutput[i + 1][j]);

        }

        OutputPoints.get(i).compressData();

    }

    // Sending the data to the database

    ProtocolCommunicator DataSender = new ProtocolCommunicator();

    if (run_type.equalsIgnoreCase("realtime")) {

        DataSender.sendToDatabase(con, OutputPoints);

    } else if (run_type.equalsIgnoreCase("backfill_insert")) {

```

```

        DataSender.sendToDatabaseBackfill(con, OutputPoints);

    }

    // Use of the API ends
    *****

} // end of main

```

```

static List<String[]> loadPointDetailsFromFile(File Point_conf_file,
        int ColumnCount) {

    List<String[]> PointDetails_arr = new ArrayList<String[]>();

    String current_line = "";

    BufferedReader in_buffer = null;

```

```

try {

    in_buffer = new BufferedReader(new FileReader(Point_conf_file));

    for (;;) {

        current_line = in_buffer.readLine();

        if ((current_line == null) || current_line.isEmpty()
            || current_line.startsWith("="))
            break;

        String[] splitted_current_line = current_line.split(",");
        PointDetails_arr.add(splitted_current_line);

    } // end of for loop

    in_buffer.close();

} catch (java.io.FileNotFoundException e) {

    e.printStackTrace();

} catch (java.io.IOException e) {

```

```
        e.printStackTrace();
    } finally {

        try {

            if (in_buffer != null) {

                in_buffer.close();

            }

        } catch (java.io.IOException e) {

            e.printStackTrace();

        }

    }

    return PointDetails_arr;
}
}
```

20 Analytics Container

The UDM Analytics container is distributed in the package called `udmAnalytics.zip`. It is a standalone java application, very similar to the UDM ProtocolCommunicators. It runs continuously and depending on the `PollInterval` (in this case the interval for the calculation) specified for the output Point(s), calls the method called “calculate” in the Java class specified by the calculation. In order to take advantage of it, the following steps have to be followed.

20.1 Build the application

There has to be a Java class in the application package that implements the calculate method defined as the following.

```
public void calculate( Object conForInputDataGen, ArrayList<Point>
inputPoints, ArrayList<Point> ThreadPoints, ArrayList<Double>
calcParamValues, String run_type, Long UTCnow )
```

<code>conForInputDataGen:</code>	this is the database connection object that is passed to the method by the container, for querying the database to fetch data for the input Points
<code>inputPoints</code>	this is the List of input Point objects
<code>ThreadPoints</code>	this is the List of output Point objects
<code>calcParamValues</code>	This List contains all the parameters to be used in the calculation. The values of the parameters are specified through a flat file, so changes in parameter values can be performed without accessing the database
<code>run_type</code>	“backfill” or “realtime”
<code>UTCnow</code>	Inside the calculation, this number should be considered as the current time. That way, the calculation will be run_type independent.

Following is a sample code that converts temperatures from degF to degC.

```
package gov.lbl.uom;

import gov.lbl.udm.DatabaseUtility;

import gov.lbl.udm.IllegalTimeFormatException;

import gov.lbl.udm.metadata.Point;

import gov.lbl.udm.timeseries.DatabaseQuery;

import gov.lbl.udm.timeseries.NoSelectedPointException;

import gov.lbl.udm.timeseries.PointData;


import java.sql.Connection;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.List;


public class DegfToDegc {


    public DegfToDegc() {


    }

}
```

```

public void calculate( Object conForInputDataGen, ArrayList<Point> inputPoints,

    ArrayList<Point> ThreadPoints, ArrayList<Double> calcParamValues, String run_type, Long
UTCnow ) {

    try {

        Connection conForInputData =(Connection) conForInputDataGen;

        // Set up variables *****

        int i = 0;

        // This portion is not needed for this particular calculation *****

        String input_start_time = "now-15m";

        String input_end_time = "now";

        DatabaseUtility Query_Utility = new DatabaseUtility();

        long[] start_end_times = Query_Utility.getLongStartEndTime

        ( input_start_time,  input_end_time,

            Calendar.getInstance().getTimeZone(),

```



```

        UTCnow);

    long start_time = start_end_times[0];

    long end_time = start_end_times[1];

    String start_BoundaryType = ">=";

    String end_BoundaryType = "<=";


    // Fetch input data *****

    DatabaseQuery DBQInstance = new DatabaseQuery();

    List<PointData> rawInputData = null;

    if ( run_type.equalsIgnoreCase("backfill") ) {

        rawInputData = DBQInstance.getDataLastStored(conForInputData, inputPoints,

            ThreadPoints.get(0).next_poll_time);

    } else {

```

```

        rawInputData = DBQInstance.getDataLastStored(conForInputData, inputPoints);

    }


    // Debug *****

    /*

    for (i = 0; i < 1; i++) {

        int j = 0;

        int count = rawInputData.get(i).Points.size();

        for ( j = 0; j < count; j++ ) {

            System.out.println("\nPoint" + i + ", Entry" + j +

                ": Timestamp --> " + rawInputData.get(i).Points.get(j).Timestamp

                + ", Value --> " + rawInputData.get(i).Points.get(j).Value);

        }

    }

    */

```

```

// Perform calculations *****

int OutputPointsCount = ThreadPoints.size();

double[] result = new double[OutputPointsCount];

for (i = 0; i < OutputPointsCount; i++) {

    result[i] = (rawInputData.get(i).Points.get(0).Value - 32.0) * (5.0/9.0);

}


// Populate UnCompressed *****

for (i = 0; i < OutputPointsCount; i++) {

ThreadPoints.get(i).UnCompressedTimestamp.add(rawInputData.get(i).Points.get(0).Timestamp);

    ThreadPoints.get(i).UnCompressedValue.add(result[i]);

}

```

```

// Debug *****

/*

for (i = 0; i < 1; i++) {

    int j = 0;

    int count = ThreadPoints.get(i).UnCompressedTimestamp.size();

    for ( j = 0; j < count; j++ ) {

        System.out.println("\nPoint" + i + ", Entry" + j +

            ": Timestamp --> " + ThreadPoints.get(i).UnCompressedTimestamp.get(j)

            + ", Value --> " + ThreadPoints.get(i).UnCompressedValue.get(j));

    }

}

*/

```

```

    } catch (IllegalTimeFormatException e) {

        // TODO Auto-generated catch block

        e.printStackTrace();

    } catch (SQLException e) {

        // TODO Auto-generated catch block

        e.printStackTrace();

    } catch (NoSelectedPointException e) {

        // Irrelevant, because this Exception cannot happen here due to all the up-front checking

        e.printStackTrace();

    }

}

}

```

20.2 Create the Points

Following are the Point details that need to be specified.

datasourceaddress1	Although this field is not used in the current version, it needs to have a value because it is a NOT NULL field
datasourcedetail1	The fully qualified name of the Java class that implements the "calculate" method
datasourcedetail2	"," separated Ids of the input Points. The order specified here will be maintained when the framework provides the Point objects as the input to the calculation.
datasourcedetail3	"," separated Ids of the output Points. The order specified here will be maintained when the framework provides the Point objects as the output to the calculation. NOTE: The Points have to be created without this field and then have to be edited, because the IDs will not be known while creating the Points.

datasourcedetail4	The name of the file containing the parameter (name=value) tuples. If there is no parameter, it can be left blank.
datasourcedetail5	Leave this blank. However, it will be automatically populated with ";" separated parameter (name=value) tuples as retrieved from the parameter file.

20.3 Edit the Points

Since DataSourceDetail3 is a necessary field, but can only be populated after the Points have been created, all the Points (the output Points, that is,) have to be updated with this information.

20.4 Application deployment

1. Package the Application (it has to be written in Java) as a jar file
2. Drop the jar file in the lib directory inside the udmAnalytics directory
3. If the jar uses some external libraries, those can be dropped in the same lib directory as well
4. Stop udmAnalytics container, if it is already running
5. Update the classpath.txt file by appending the only non-empty line
6. Run the script called udmAddCalculation.bat (Windows) or udmAddCalculation.sh (linux)
7. Start udmAnalytics container
8. Typically the realtime instance of udmAnalytics will run continuously, while the backfill mode instance will be run only in batch mode. Two instances can run simultaneously. While running in backfill mode, the list of output Points will have to be provided in a csv file.

21 Acknowledgements

Hereby, we showcase the projects and the corresponding PIs that showed the strategic vision to provide funding for the development of this tool as part of their projects:

- Integrated Systems User Test Bed Facility and National Testing Network project funded by DOE and led by Steve Selkowitz
- DER-CAM Software as a Service (SaaS) Commercialization project funded by DOE TCF program (with matching in kind support from OSIssoft LLC), done for a site at UC Davis and led by Chris Marnay
- "CERTS Energy Manager" funded by the "Office of Electricity Delivery and Energy Reliability" at DOE, led by Chris Marnay
- Automated Continuous Commissioning project funded by DoD ESTCP program being done for a site located at Great Lakes, IL and led by Phil Haves
- Characterization of HVAC Equipment Responsiveness and Field Performance Assessment Framework, funded by DOE and led by Mary Ann Piette
- Demo and Characterization of Data Center Cooling Equipment, funded by CEC and led by William Tschudi and Henry Coles
- Demo and Characterization of Controls of VFD Driven Cooling Equipment for Data Center Cooling, funded by CEC and led by William Tschudi and Henry Coles
- Simulation-Based Continuous Commissioning performed at NASA, Ames Sustainability Base, funded by DOE through the CBP program and led by Cindy Regnier and Kristen Parrish

22 References and Software Libraries

1. <http://www.epa.gov/greenbuilding/pubs/gbstats.pdf>
2. http://www.brookings.edu/~media/Files/rc/papers/2008/05_carbon_footprint_sarzynski/carbonfootprint_brief.pdf
3. Joda-time (<http://joda-time.sourceforge.net/>) - Java library for date and time manipulation
4. Apache commons math (http://commons.apache.org/math/download_math.cgi) – Java mathematics library
5. PostgreSQL JDBC4 driver (<http://jdbc.postgresql.org/download.html>)
6. Jamod (<http://jamod.sourceforge.net/>) – Modbus library in Java
7. Ipmiutil (<http://ipmiutil.sourceforge.net/>) – IPMI Management Utilities
8. JFreeChart (<http://www.jfree.org/jfreechart/>) – Java based charting library

23 Appendix I: Ideal TDMS Features

23.1 Platform and browser support

It's probably not necessary for the core components (middleware and database) of the software to support both Windows and linux, but it would be nice. From that standpoint, use of Java as the programming platform seems reasonable.

It is absolutely necessary for it to support all major web browsers.

23.2 Database

The system should be as standards-oriented as possible so that the support ecosystem for the system is large. That guides us to build this system to use SQL standard based database. Ideally, it should be compatible with all major SQL database systems such as Oracle, PostgreSQL, DB2, MySQL and MS SQL Server.

23.3 DataType support

The system should support at least the following data types.

- 64 bit floating point (double precision)
- 32 bit integer
- Binary Large Object (BLOB) – this data type is required to store images both for metadata and for time series coming out of monitoring camera.
- String

23.4 Data collection

There should be file download and file unzip capabilities.

The tool should support the following **protocols** for data collection:

- XML parsing
- Structured text file parsing
- SQL-based database query
- BACNet
- Modbus
- DDE (Dynamic Data Exchange)
- DNP 3

- SNMP
- IPMI

There should be backfill mode of data collection to recover history.

23.5 Time stamp

- Timestamp should be portable across time zones.
- Difference between device clock and server clock should be identified and correction should be applied automatically.

23.6 Metadata

Source of metadata

- Provision for manual entry should be there. However:
- It should nicely integrate with the Configuration Management Database (CMDB) to pick up information contained in there.
- It should communicate with the electronic design document of the facility to automatically gather information on structural, mechanical, electrical, plumbing, ducting etc.
- For applications within the building industry, that calls for compatibility with Building Information Model (BIM), specifically Industry Foundation Classes (IFC)

23.7 Navigation

The navigation through all the points should be easy and intuitive. It should not expect the user to remember what BAC.OAK.CC.1 means. That was just an example from typical practice of BAS system integrators (SI). The point is that the metadata should be extensive but lucid. More details on this can be found in section 23.12.

23.8 Data model

The data model of the metadata will need to contain context (for example, the location that this data pertains to), quantity, data source type (measured, model1, model2 etc.), any asset associated with the location, asset history of the location (if assets were moved out or replaced), history of a single asset (for example, maintenance history)

23.9 Security

There should be a per-user based security for both metadata and time series data. Also, there should be time-sliced security for a single time-series point.

23.10 Data query

There has to be data clean-up/interpolation options.

23.11 Horizontal Scalability

It should be **scalable in terms of number of points**. There should not be any limit as long as we can provide commodity grade hardware (NOT a \$100,000 cluster). Essentially, it should be able to scale out

easily (horizontal scalability). At the same time, it would be nice to have the result of a benchmark test to know the approximate number of points that a specific hardware can handle with reasonable performance.

It should be **scalable in terms of data throughput**. Consider, collecting 600,000 points every 30 s (We do not necessarily need to store all of them.). That's 20,000 points/s distributed over the time period. The system should be able to scale just by the addition of commodity grade hardware into the system.

23.12 User Interface

- The visualization should be **web based**.
- It should have 2D and 3D views; zoom and pan capabilities; its search for location, asset, quantity, point etc should be like google maps, i.e. **easy but informative**; there should be capability to superimpose time-series data on the metadata (location, asset etc.); not to mention plain and simple time-series displays such as trend and dial gauge should be there.
- There has to be a way to upload graphic and then to superimpose numbers on that. This could be very useful in putting numbers on a floor layout or campus map in the absence of design data or even in other conditions.
- There should be buttons on the screens for sending manual signals to controllers or analysis engines.

23.13 Extensibility

23.13.1 Analytics

The system should be extensible, that is, it should have an API so that anybody can build his/her own analysis that interacts with this system. The native API is faster and can be used by applications developed in the native programming environment. There should be a standards-based API, such XML SOAP, so that non-native applications can use this system as well.

23.13.2 Visualization

Similar is true for displays, that the tools and components should be available so that users can build new display screens easily.

23.14 Data compression

The tool should have **lossless data compression** mechanism. We want to keep everything forever, but in an efficient manner. Some tools average the data as the data gets old. That prevents retroactive analysis. The data compression should be only on the streaming data, so that data retrieved from the database at any time would have repeatability.

23.15 Ease of deployment

The time required for configuring the system for a particular site should be as little as possible. Built-in templates for various equipment and auto-discovery of IP enabled devices help reduce deployment time.

23.16 Redundancy

The system should support redundancy, which is necessary for high availability and load balancing.

23.17 Forecast data

Forecast data has some different characteristics than historic monitored data. The system should be able to handle forecast data.

23.18 Change management

It should have the capability to propagate changes made at one part of the system to all the parts of the system that are affected by that change.

It should have capability to pick up changes in external systems such as the CMDB or the design document, and propagate that change through the whole ecosystem.